Oracle Utilities Network Management System

Configuration Guide

Release 2.5.0.1.0

F41248-02

March 2021

ORACLE

Oracle Utilities Network Management System Configuration Guide, Release 2.5.0.1.0

F41248-02

Document Revised April 2021.

Copyright © 1991, 2021 Oracle and/or its affiliates. All rights reserved.

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, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are “commercial computer software” or “commercial computer software documentation” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government’s use of Oracle cloud services are defined by the applicable contract for such services. 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 Inside 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, Epyc, and the AMD 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 about 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 unless otherwise set forth in an applicable agreement between you and Oracle. 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, except as set forth in an applicable agreement between you and Oracle.

Preface 1-xv
Audience 1-xv
Related Documents 1-xv
Conventions 1-xvi
System Overview 1-1
System Overview 1-1
User Environments 1-2
Isis 1-3
Database 1-3
Hardware and Third Party Software 1-4
Network Architecture 1-4
Architecture Guidelines 1-4
Overview 1-4
Product Dependencies and Locations 1-5
Oracle Utilities Network Management System High-Level Conceptual Diagram 1-8
Example Hardware/Software Overview 1-9
Hardware Sizing 1-10
Standard Product Implementation 2-1
Overview 2-1
Software Release Level 2-2
Installation 2-2
Interfaces 2-2
Modeling and GIS Integration 2-2
GIS Model Extractor 2-2
Standard Preprocessor 2-3
Device Types and Attributes 2-3
Software Configuration Dependencies On Device Types 2-3
Operations Modules Software Configuration 2-4
Overview 2-4
Web Workspace 2-5
Web Trouble 2-6
Web Call Entry 2-7
Web Callbacks 2-7
Web Switching Management 2-7
Power Flow 2-8
Fault Location Analysis (FLA) 2-8
Fault Location, Isolation, and Service Restoration (FLISR) 2-8
Feeder Load Management (FLM) 2-8
Suggested Switching 2-8
Optimization 2-9
Load Shed and Restoration 2-9
SCADA Extensions 2-9
Service Alert 2-9
Storm Management 2-9
Isis Configuration 3-1
Isis Terminology 3-2
Isis Architecture 3-2
Isis Directory Structure 3-3
run_isis 3-3

Isis Configuration Files ... 3-4
sites File ... 3-4
isis.rc Startup File ... 3-4
Isis Environment Variables ... 3-5
ISISPORT and ISISREMOTE ... 3-5
ISIS_PARAMETERS ... 3-5
Isis Standalone Mode ... 3-6
Isis Multi-Environment Considerations ... 3-6
Isis Log Files ... 3-6
isis.[date].[time].log ... 3-6
[Site No.].logdir ... 3-6
The Protos Log ... 3-6
The Incarn Log ... 3-6
Starting Isis ... 3-7
nms-isis ... 3-7
Initializing Isis ... 3-7
Stopping Isis ... 3-7
Starting Isis on Non-Default Ports ... 3-8
The cmd Tool ... 3-8
Exiting cmd ... 3-9
Troubleshooting ... 3-9
Generating an Isis Dump File ... 3-9
Generating an Isis Dump File for All Applications ... 3-10
Reporting a Problem to Customer Support ... 3-10
Information Lifecycle Management ... 4-1
Prerequisite ... 4-1
History Tables ... 4-2
Database Configuration ... 5-1
Oracle Installation Guidelines ... 5-1
Oracle Tablespaces ... 5-1
Oracle Instances ... 5-2
Other Environment Variables ... 5-3
Oracle Users ... 5-4
Security Roles ... 5-4
Users ... 5-4
Starting Oracle ... 5-5
Environment Configuration ... 6-1
CES_PARAMETERS ... 6-1
CES_PARAMETERS Schema ... 6-1
Site-specific Parameters ... 6-2
Environment-Specific Parameters ... 6-2
NMS_PARAMETERS_VIEW ... 6-2
Encrypting Configuration Parameters ... 6-3
Encrypting Passwords with Oracle WebLogic Server Utility ... 6-3
The System Resource File ... 6-4
Modifying Environment Variables ... 6-4
nms-env-config ... 6-4
Environment Variables ... 6-5
Dual-Environment Configuration ... 6-10

Services Configuration 7-1
Services Overview 7-1
SMService - System Monitor Service 7-2
DBService - Database Service 7-3
ODService - Object Directory Service 7-3
DDService - Dynamic Data Service 7-4
MTService - Managed Topology Service 7-4
JMService - Job Management Service 7-5
TCDBService - Trouble Call Database Service 7-5
MBService - Model Build Service 7-5
SwService - Switching Service 7-5
MBDBService - Model Build Database Service 7-5
MQDBService - MQService Gateway DBService 7-6
PFService - Power Flow Service 7-6
CORBA Gateway Service 7-6
Service Alert Service 7-7
Service Alert Email Administration 7-9
How Service Alert Email and Paging Notification Work 7-9
Entering Email/Pager Configuration Settings 7-9
Email Subject Line Configuration 7-10
Service Alert Script Configuration 7-12
Services Configuration File 7-13
Scripts 7-13
Server 7-16
Service 7-16
Program 7-18
displayName 7-19
Instance 7-19
Model Build System Data File 7-20
Starting and Stopping Services 7-20
Starting Services 7-20
Stopping Services 7-21
Starting and Stopping the WebLogic Managed Server 7-22
Building the System Data Model 8-1
Model Builder Overview 8-2
Patches 8-2
Data Directories 8-3
OPERATIONS_MODELS Directory 8-3
Model Configuration 8-4
Define Environment Variables 8-4
Configure Isis 8-6
Verify Database Connection 8-6
Directory Set Up 8-7
Define and Organize Classes 8-7
Configure Attribute Table 8-8
Configure Control Zones 8-8
Configure Symbology 8-9
Service Configuration File 8-9
Verify Licensed Products File 8-9
Run Automated Setup 8-10

Linking In Customers 8-14
Customer Model - Logical Data Model 8-15
Residential Model 8-16
Commercial and Industrial (C & I) Model 8-17
Customer Model Database Schemas 8-18
Customer Model Database Tables 8-18
Customers Table 8-19
Service Locations Table 8-21
Meters Table 8-24
Account Type Table 8-25
Service Points Table 8-25
DERs Table 8-26
Linkages to Other Tables 8-27
Customer Model Views 8-28
CES Customers View 8-28
Customer Sum View 8-31
Model Build Process 8-33
Model Build with a Preprocessor 8-33
Model Build with a Post-Processor 8-34
The Model Build Preprocessor 8-35
Model Build Basics 8-36
Model Preprocessor 8-37
Format for the Explosion Definition File 8-50
Example of Cell Definitions 8-53
Model Build Workbooks 8-55
Model Build Process for Work Orders 8-67
Power Flow Engineering Data Workbook 8-67
Load Shed and Restoration 8-71
Model Manipulation Applications and Scripts 8-73
DBCleanup 8-73
nms-delete-map 8-74
nms-delete-object 8-74
nms-delete-patch 8-75
AuditLog 8-75
Schematics 8-76
Model Requirements for Schematics 8-76
Schematic Limitations 8-76
Configuring Schematics 8-76
Schematic-Specific Symbology 8-86
Generating Schematics 8-86
The Post-Build Process 8-86
Creating the Import Files 8-86
Processing the Import Files 8-86
Aggregate Devices 8-87
Model Requirements for Aggregate Devices 8-87
In Construction Pending / Device Decommissioning (ICP) 8-87
Device Lifecycles 8-87
Model Requirements for ICP 8-88
Model Builds and Commissioned/Decommissioned Devices 8-88
Effect of ICP Devices on Network Topology 8-88

ICP Device Symbology 8-89
Troubleshooting Issues with ICP Device Symbology 8-89
Auto Throw-Over Switch Configuration (ATO) 8-90
Model Requirements for ATOs 8-90
Summary Object Configuration 8-91
Adding Latitude and Longitude Attributes to Objects in the Model Build Process 8-93
Coordinate Systems 8-94
Geographic Coordinate System 8-94
Quasi-Geographic Coordinate System 8-94
Internal World Coordinate Systems 8-94
Overview Schematic Coordinate Systems 8-95
One Line Schematic Coordinate Systems 8-95
Latitude and Longitude Coordinate Systems 8-95
Symbology 8-96
Firm Symbols 8-96
Hard Symbols 8-102
1D Width Multiplier 8-103
Soft Symbol Definitions 8-103
Pixmap Symbols 8-111
SVG Symbols 8-112
SVG Master Symbol Process 8-118
Converting SYM Files to SVG 8-123
SVG Performance Tool 8-124
Updating Symbology 8-126
Symbology Mapping 8-127
The SYMBOLOGY_STATE Table 8-127
The QUALITY_RULES Table 8-128
The CONDITION_RULES Table 8-128
DMS Data Model Requirements and Configuration 8-129
Data Import and Build Process 8-129
kVA Solutions 8-129
Modeling Device Data 8-130
Sources 8-130
Line Impedances 8-130
Transformers and Regulators 8-131
Distributed Generation 8-131
Capacitors and Reactors 8-131
Switch Ratings 8-132
Modeling Loads 8-132
DER and the Weather Zone Forecast 8-135
Bellwether Meter Voltages 8-142
Viewing Power Flow Attributes 8-142
Power Flow Service High Level Messages and Command Line Options 8-143
Power Flow Service High Level Messages 8-143
Power Flow Service Command Line Options 8-145
Spatially Enabling the Data Model for Outage Analytics 8-146
NMS CIM Import and Export Tools 8-147
CIM Import 8-147
CIM Export 8-147
Preparing the NMS Model for Oracle Utilities Customer Self Service 8-148

Materialized Views ... 8-148
Adding Outage Summary Spatial Landbase to the NMS Viewer ... 8-151
Model Build File Export to XML ... 8-156
MB Export to XML ... 8-156
Schematic Data Export to XML ... 8-156
Database Maintenance ... 9-1
Oracle Configuration ... 9-1
Indexes ... 9-1
Generating Statistics ... 9-2
Make Tablespaces Locally Managed ... 9-3
Block Size ... 9-3
Compatibility ... 9-3
Site Guard ... 10-1
Prerequisites ... 10-1
Site Installation and Configuration ... 10-2
Custom Scripts ... 10-2
Custom Scripts for Database Server ... 10-2
Custom Scripts for NMS server ... 10-3
Other Custom Scripts ... 10-3
Configuring Custom Scripts ... 10-4
Operation Plans ... 10-4
Start Operation ... 10-4
Stop Operation ... 10-5
Switchover Operation ... 10-5
Failover Operation ... 10-6
Failover Patching Operation ... 10-6
Additional NMS Configuration ... 10-7
Read Only Mode ... 10-7
Syncing Operations Model ... 10-8
nms-wls-config ... 10-8
High Availability Configuration ... 11-1
Overview ... 11-1
NMS Agent ... 11-1
NMS Monitor ... 11-1
CESEJB, NMS-WS ... 11-1
WebLogic Server ... 11-1
Database Server ... 11-2
EMCLI ... 11-2
Site Guard ... 11-2
ZooKeeper ... 11-2
Architectural Overview ... 11-3
Components at One NMS Site ... 11-3
Components at Three Sites ... 11-4
Recommended Configuration Order ... 11-5
EM CLI Standard ... 11-8
Download ... 11-8
Installation ... 11-9
Setup ... 11-9
Configuring NMS Monitor ... 11-9
ZooKeeper ... 11-10

Keystore Creation 11-10
Export the Certificate 11-10
Creating the Truststores 11-11
Environment 11-11
ZooKeeper Configuration File 11-12
Node Id 11-12
ZooKeeper Data Directory 11-13
Starting and Stopping the Node 11-13
NMS Monitor 11-13
Overview 11-13
Configuration 11-14
Setting the Domain Environment 11-14
Configuration Tool Main Page 11-15
Configure ZooKeeper 11-16
Configure Datasource 11-17
Configure emcli 11-18
Keystore Actions 11-19
Truststore Actions 11-20
Credentials 11-21
Creating the NMS Monitor Server 11-23
NMS CESEJB Self-Signed Certificates 11-25
NMS Agent 11-25
NMS Agent Overview 11-25
Configuration 11-26
Administration User Profile 11-26
Configuration Tool Main Page 11-27
Configure Server 11-27
Keystore Actions 11-28
Generate Service Script 11-30
The nmsagent Directory 11-32
NMS Monitor Web Application 11-32
Launching NMS Monitor and Logging In 11-32
NMS Monitor Main Screen 11-33
NMS Site Status Screen 11-33
Possible Status Values 11-34
Manual Failover 11-35
Configuration Screen 11-36
Sites 11-36
Automatic Failover 11-38
Failover Priority 11-39
Failover Operation Plans 11-40
Troubleshooting and Support 12-1
Troubleshooting an Issue 12-1
Evaluating System Status 12-1
Examining Log Files 12-1
Examining Core Files 12-9
Identifying Memory Leaks with monitor-ps-sizes 12-11
Validating the WebLogic Caches with NMS Services 12-12
Monitoring EclipseLink Related Database Transactions 12-12
Performance Testing 12-13

Other Troubleshooting Utilities 12-15
Oracle Support Information 12-16
Support Knowledgebase 12-16
Contacting Oracle Support 12-16
NMS Data Quality 13-1
Setting Up NMS for Oracle Utilities Analytics 14-1
User Authentication 15-1
Overview of Authentication 15-2
Configuring the WebLogic Security Realm 15-2
Configuring Authentication Using WebLogic Internal Users/Groups 15-3
Configuring Authentication Using an Active Directory Provider 15-4
Configuring Authentication Using an OpenLDAP Provider 15-5
Two-Factor Authentication 15-6
Fault Location, Isolation, and Service Restoration Administration 16-1
Introduction 16-1
Fault Location, Isolation, and Service Restoration Timeline 16-2
Software Architecture Overview 16-4
Configuring Classes and Inheritance 16-6
Single Phase FLISR Operation 16-6
Restoring Substation Bus on Voltage Loss 16-6
Modeling Information 16-7
SRS Rules 16-10
High Level Messages 16-10
Troubleshooting 16-10
Distribution Management Application Configuration 17-1
Configuring Power Flow 17-1
PFService (Power Flow Service) 17-1
Non-Converged Islands 17-2
Power Flow Inheritance 17-2
Power Flow Rules 17-3
Feeder Load Management History and Accuracy 17-3
Java Application Configuration 18-1
Understanding the NMS Java Application Configuration Process 18-1
Understanding NMS Java Application Configuration Files 18-1
Understanding the Process for Building and Deploying Custom Configuration 18-2
Testing the Java Client Configuration 18-6
JBot Application Configuration 18-8
JBot Configuration Overview 18-8
Understanding the JBot XML Schema and XML Files 18-11
GUI Configuration 18-20
Advanced Configuration Options 18-22
JBot DataStore Reference 18-30
NMS JBot Tool Configuration 18-31
User Permissions 18-31
User Type Configuration 18-33
Login Tool Configuration 18-37
Master Window Configuration (Oracle Fusion Client Platform) 18-38
Java Web Start Page 18-39
Table Export Configuration 18-39

Work Agenda Configuration 18-40
Crew Actions Configuration 18-42
Event Details Configuration 18-43
Update Events Configuration 18-45
Trouble Summary Configuration 18-48
Viewer Configuration 18-49
Trending Graphs Configuration 18-57
Feeder Load Management Configuration 18-58
Model Management Application Configuration 18-59
DMS Summary Tool Configuration Guide 18-60
Right-To-Left Language Configuration 18-62
Customizing Applications 18-63
Customization Examples using the Demo Tool 18-63
Creating Custom Functions for Displaying Data 18-66
Using Additional Libraries 18-67
Invoking Commands from an External System 18-67
Invoking Commands Using a Web Service 18-68
Communicating with Other Programs Using Named Pipes 18-68
Messages Available Using the OSI SCADA Adapter 18-69
Messages from SCADA to NMS 18-69
Messages from NMS to SCADA 18-69
JBotCommand Methods Reference 18-70
JMService Configuration 19-1
Trouble Code Configuration 19-1
SRS_TROUBLE_CODES Schema 19-1
Configuration 19-2
SRS Clues and Priorities 19-4
What are Clues? 19-4
The Total Priority of a Trouble Call 19-4
The Priority of a Trouble Call 19-4
The Total Priority of an Event 19-5
The Condition Status of a Trouble Call 19-5
What Calls Are Priority Calls? 19-5
Call Quality Configuration 19-6
Schema 19-6
Set Types 19-6
Generic Event Fields 19-9
Configure Generic Fields 19-9
Modify JOBS Table 19-10
Add Columns to Work Agenda 19-10
Weighted Customer Count 19-11
Introduction 19-11
Configuration 19-11
Appointments 19-12
Appointment Types 19-12
Dispatch Groups Configuration 19-12
Tables 19-12
Damage Assessment Configuration 20-1
Schema 20-1
DAMAGE_REPAIR_TIMES 20-1

Repair Times 20-1
Default Work Queue 20-2
Examples 20-2
Damaged Asset Status Options 20-2
Example 20-2
Damaged Asset Condition Options 20-3
Example 20-3
'Road Blocked' Damaged Asset Type 20-3
Control Tool Configuration 21-1
Overview 21-1
Control Tool Configuration 21-1
Control Tool Database Table Configuration 21-1
The Control.xml File 21-5
Project_Control_Actions.inc Include File 21-9
Using the Control Config Generator 21-10
Configuration Example: Adding an Undo Close Action 21-11
Control Tool Options in the Viewer Context Menu 21-12
Updating Control Tool Configuration in Production Systems 21-12
Adding New Device Classes 21-12
Mapping Existing Actions to Existing Device Classes 21-12
Adding New Actions 21-12
Changing When Actions are Enabled 21-13
Aggregate, Secondary, or Associated Devices 21-13
Customer Counts Lists in the Look Ahead 21-13
Configurable Device Lists in the Look Ahead 21-13
Web Switching Management Configuration 22-1
Configuring Classes and Inheritance 22-2
Database Data Tables 22-4
Database Configuration Tables 22-6
Configuring Project-Specific Columns 22-7
Web Switching/Web Safety Crew Columns 22-9
Global Web Switching Parameters 22-10
SwmanParameters.properties 22-10
Note Concerning Control Action Descriptions 22-22
GUI Configuration Overview 22-23
Web Switching 22-23
Web Safety 22-25
Switching Sheets 22-27
Automated Emails 22-27
Sheet Types 22-29
State Transitions 22-29
Sheet Data Fields 22-30
Open Switching Sheet List 22-31
New Switching Sheet List 22-31
Search Switching Sheet 22-31
Device to Sheet Operation List 22-32
Model Verification 22-32
Default Crews 22-33
Versioning 22-33
Overlaps 22-34

External Documents 22-34
Generate Isolation Steps 22-34
Generic Tables 22-35
Switching Steps 22-37
Manual Step Addition 22-37
State Transitions 22-38
Control Tool Actions 22-38
Step Columns 22-38
SCADA Auto-Transitioning 22-41
View Areas 22-41
Toggling Study Mode 22-41
Step Order Execution Rules 22-41
Instructed Actions 22-42
Configuring the Sensitivity of Step Execution Buttons 22-42
Switching Sheet Email Attachment Configuration 22-43
Automatic Crew Population Option for Steps 22-43
Web Safety 22-44
State Transitions 22-44
Safety Document Data Fields 22-45
Search Safety Document 22-46
Configuring Stand Alone Safety Documents 22-46
High Level Messages 22-47
SwService 22-47
Web Switching and Web Safety 22-48
Troubleshooting 22-49
Summary of Java Action Commands 22-50
NMS Training Simulator Configuration 23-1
Environment Configuration 23-1
Power Flow Emulator Configuration 23-2
EMService High Level Messages 23-2
JBot Configuration 23-3
General User Environment 23-3
Switching 23-3
Control Tool 23-3
SRS Rules 23-3
NMS Flex Operations Configuration 24-1
Overriding Configuration and NLS 24-1
Flex Operations and OMA Shared Configuration 24-2
Overriding Configuration 24-2
Functions 24-3
Observables 24-3
Merging or Replacing Array/Object Content 24-3
Arrays 24-4
Overriding NLS 24-5
Common Terms/Concepts 24-6
Note on Grammar Issues 24-6
Layout Configuration 24-7
Custom Fields 24-9
Enabled/Visible Configuration 24-10

Oracle BI Publisher Reports Configuration 25-1
CES_PARAMETERS Configuration for BI Publisher Reports 25-1
Installing the Oracle BI Publisher Report Packages 25-2
Installation 25-2
Altering and/or Translating the Reports 25-4
Adding XLIFF translation file 25-4
Updating the Sub-Template and Template Files 25-4
Updating the Report Template File 25-5
Changing Date Formats 25-6
Contents of the WebSwitching Folder 25-6
Contents of the WebSwitchingImpactedCustomers Folder 25-7
Contents of the WorkAgenda Folder 25-7
Building Custom Applications 26-1
Overview 26-1
Prerequisites 26-2
Compiling C++ Code Using the Software Development Kit 26-3
Building Sample AMR and AVL Adapter 26-5
Build Instructions 26-5
Deployment 26-5
OMS for Water 27-1
Command Line Options and High Level Messages A-1
Command Line Options A-1
Command-Line Options Valid For All Services A-1
DBService A-2
DDService A-2
JMService A-6
MBService A-7
MTService A-9
ODService A-10
High Level Messages A-11
Action Commands Supported By All Services A-11
DDService A-13
JMService A-14
MBService A-17
MTService A-17
ODService A-17
Model Edit Configuration B-1

Preface

Please read through this document thoroughly before beginning your product implementation. The purpose of this guide is to provide implementation guidelines for a standard Oracle Utilities Network Management System implementation. This document discusses installation, interfaces, modeling, and software configuration that are considered typical and acceptable for a standard product implementation.

Audience

This document is intended for anyone responsible for the implementation of Oracle Utilities Network Management System.

Related Documents

- *Oracle Utilities Network Management System Adapters Guide*
- *Oracle Utilities Network Management System Advanced Distribution Management System Implementation Guide*
- *Oracle Utilities Network Management System for Water User's Guide*
- *Oracle Utilities Network Management System Installation Guide*
- *Oracle Utilities Network Management System Licensing Information User Manual*
- *Oracle Utilities Network Management System Operations Mobile Application Installation and Deployment Guide*
- *Oracle Utilities Network Management System Quick Install Guide*
- *Oracle Utilities Network Management System Release Notes*
- *Oracle Utilities Network Management System Security Guide*
- *Oracle Utilities Network Management System User's Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Chapter 1

System Overview

The Oracle Utilities Network Management System is an operations resource management system that runs on a Unix/Linux platform. The system administrator is responsible for maintaining the Unix operating system, the Oracle Utilities Network Management System, and the PC connections to remote workstations. This guide provides details about installing, optimizing, and troubleshooting the Oracle Utilities Network Management System and assumes that the reader is an experienced Unix/Linux user.

- System Overview
- Hardware and Third Party Software
- Network Architecture
- Architecture Guidelines

System Overview

An Oracle Utilities Network Management System includes:

- Isis internal message bus
- Oracle Utilities Network Management System services
- Oracle WebLogic Java Application Server
- Java Oracle Utilities Network Management System Clients
- Oracle Relational Database Management System

The Oracle Utilities Network Management System can be broken down into individual components. Each component is installed and configured separately. Oracle Utilities Network Management System uses a client/server architecture. The server supports Oracle Utilities Network Management System daemon processes, while the clients display a graphical user interface to allow the user to interact with the system. Internal daemon service process to daemon service process communication is managed with a concurrency management and messaging system called Isis. Isis is the backbone of the communication architecture for an Oracle Utilities Network Management System. The network model, system configuration, and operational data is all stored persistently in an Oracle database.

The table below describes the Oracle Utilities Network Management System components.

Component	Description
Client User Environments	The Java-based end-user environments are configured using a combination of SQL files (RDBMS table based configuration), XML files, and Java properties files. The XML files are based on an NMS-specific XML schema, which provides the foundation for Java user interface customization.
Isis	Clients access services and tools through a central concurrency management and messaging system called Isis. Isis is a real-time implementation of message oriented middleware that helps provide access to the Oracle Utilities Network Management System daemon service processes as well as inter-daemon process communication.
Services	Services maintain and manage the real-time electrical network data model. Services also cache information from the database tables to optimize client information access.
Oracle WebLogic	Oracle WebLogic hosts Oracle Utilities Network Management System specific Enterprise Java Beans (EJBs). These EJBs help cache the network model and process updates/requests to/from Java clients as well as to/from external systems.
Web-Gateway	The Web-Gateway is a CORBA (Common Object Request Broker Architecture) interface between Network Management System daemon processes and WebLogic EJBs.
Oracle Database	The Oracle Database contains the complete network data model, configuration, and operational data history of an Oracle Utilities Network Management System.

Note: Services, applications, and the Oracle RDBMS tablespaces can be spread over multiple servers or run on a single server. The simplest configuration is for everything (Oracle RDBMS, Oracle Utilities Network Management System services and Oracle WebLogic Java Application Server to run on a single (generally SMP) server. Common variations would include the use of a cluster based hardware server to support high-availability (for Oracle RDBMS and Oracle Utilities Network Management System Services). This provides flexibility for system configuration, depending on your needs and hardware.

User Environments

Oracle Utilities Network Management System provides distinct user environments based on the tasks that different users perform. The system configuration defines the relationship between a user type and the grouping of tools that make up their user environment. The user type defines not only what tools are visible when their particular environment opens, but also what tools and modes of operation are available to that user.

The Java/Swing-based end-user environments are configured using a combination of SQL files (RDBMS table based configuration), XML files, and Java properties files. The XML files are based on an Oracle Utilities Network Management System XML schema that allows the Java user interface to be customized for a particular project implementation.

Isis

Isis is a synchronous/asynchronous real-time publication/subscription message bus. Isis is used to coordinate requests and updates between Oracle Utilities Network Management System daemon processes. Isis only runs on the same node as Oracle Utilities Network Management System daemon processes.

Database

Oracle Utilities Network Management System requires an Oracle relational database management system (RDBMS). The database persistently manages the tables that define the information constructs of the electrical network data model (sometimes called an operations model). Oracle Utilities Network Management System services cache information from the relational tables. These tables include the management of system constructs such as handles and aliases, class hierarchy, topology model, device status, events, incidents (trouble calls), outages, alarms, switch plans, and conditions.

Database installation and configuration follow these basic steps:

- The Oracle RDBMS is initially configured using the product's standard installation and configuration procedures. To help you get started, Oracle provides an example network data and customer model (Oracle Power and Light) that can be installed out of the box.
- Using Oracle Utilities Network Management System utilities, the initial schema is installed and populated. For a new project installation (not out of the box Oracle Power and Light), note that significant work must generally be undertaken to translate available electrical network topology and customer model data into the standard schema required by the Oracle Utilities Network Management System. This is an effort often measured in weeks or months, not days. Proper conversion of available network and customer data to the standard Oracle Utilities Network Management System schema is generally the most time consuming aspect of a project implementation.
- If you are performing an upgrade, you may need to perform a schema migration and/or an RDBMS based configuration migration.

All Oracle Utilities Network Management System schema definitions follow the SQL standard. Schema installation and population use SQL scripts that are generally executed via the SQL interface (ISQL) to the Oracle RDBMS instance. The necessary data elements required for an Oracle Utilities Network Management System consist of the following components.

Component	Description
Oracle Tablespaces	Used for persistent storage of production data (e.g., network components, operations data, etc), customer information and indexes. The Oracle Utilities Network Management System model is typically loaded into two or more separate tablespaces, Electrical Network Operations data, and Customer Model data (name, address, phone, account, etc.).

Hardware and Third Party Software

Since specific system requirements can change with new releases, they are not available as part of this document. For the most current requirements, refer to the *Oracle Utilities Network Management System Quick Install Guide.* Information on de-supported platforms and deprecated integrations is provided in the *Oracle Utilities Network Management System Release Notes*.

Network Architecture

Running Oracle Utilities Network Management System software over a shared local area network and wide area network requires a network analysis. Network latency can cause significant problems with an Oracle Utilities Network Management System. Specifically care should be taken to ensure adequate bandwidth and minimal latency between all major Network Management system back end components. This includes the Network Management System node, the Oracle RDBMS node and the Oracle WebLogic application server node.

In addition, if you intend to support Oracle Network Management System clients over a high-latency or low bandwidth connection, non-standard deployment schemes may be necessary to ensure acceptable client performance. For example it may be required to run the Oracle Network Management System Java clients on something similar to a Windows Remote Desktop Services type infrastructure.

Architecture Guidelines

This chapter provides an overview of the product module dependencies and locations, the logical hardware relationships, and sample physical hardware implementations:

- Product dependencies and locations
- Logical hardware design
- Sample server implementations
- Hardware sizing
- Printing

Overview

The guidelines in this section complement the information contained in the Oracle Utilities Network Management System Release Notes. The Product Summary and Dependencies document has been replaced by a combination of the Release Notes and this Architecture Guidelines document.

This section contains information about product module dependencies and locations, the logical hardware relationships, and sample physical hardware implementations. It should provide the information needed to understand the relationships between the software modules and the hardware that is required to implement.

For an overall product summary, please refer to the *Oracle Utilities Network Management System User Guide*.

Product Dependencies and Locations

The following table describes Oracle Utilities Network Management System product module dependencies and their locations.

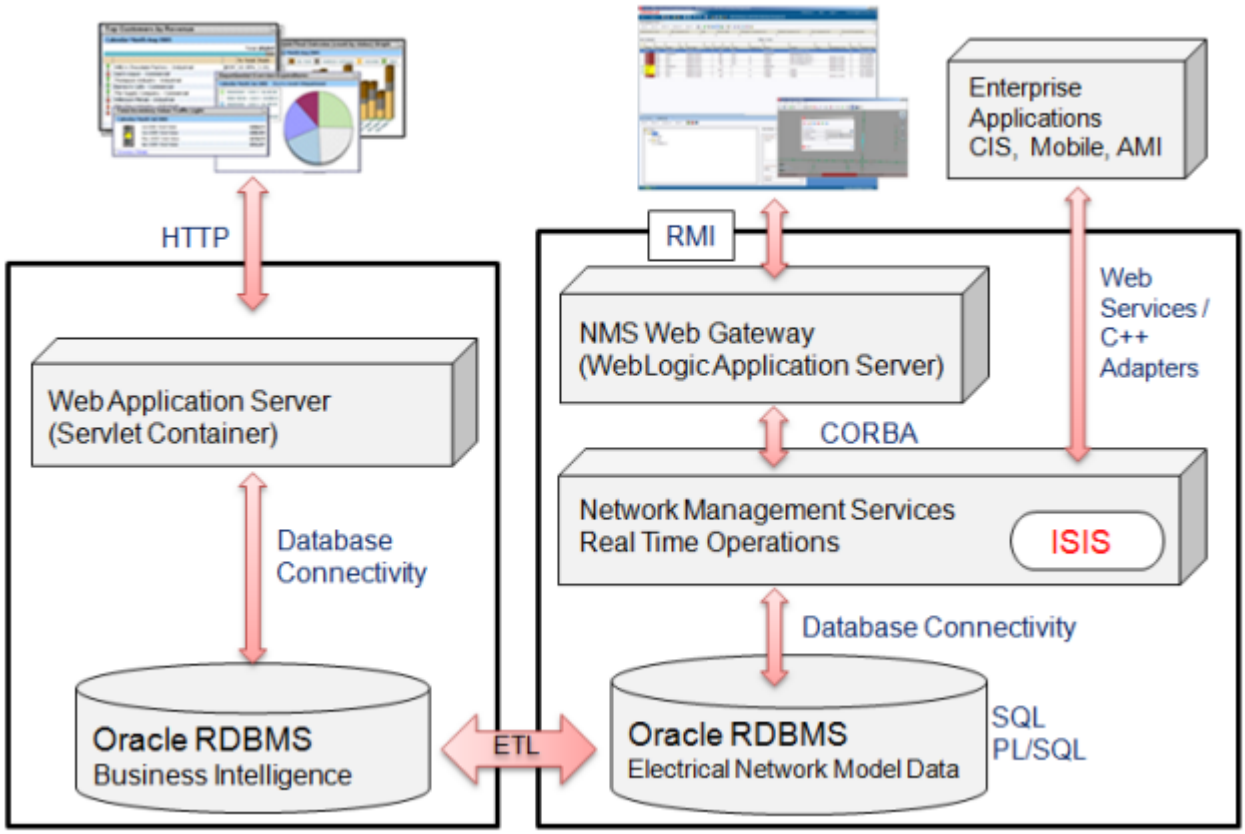
Module/ Component	Product	Dependency	Server	Client	Location
Model Management	OMS Base / DMS Base				
NMS Core Services			Unix		System Server
Web Gateway			Unix		System Server
Configuration Assistant				Windows	Web Client
US Electric Ops Model	OMS Base / DMS Base	Model Management			
Model Builder			Unix		System Server
US Standard Configuration	OMS Base / DMS Base	Model Management			
High Availability	OMS Base / DMS Base	Model Management			
Cluster Capability			Unix		RDBMS Server/ System Server
GIS Adapters	OMS Base / DMS Base	Model Management			
ESRI Adapter					GIS Server
Intergraph Adapter					GIS Server
Smallworld Adapter					GIS Server
Generic Adapters	OMS Base				
IVR Adapter		Web Trouble	Unix		System Server
CIS Adapter		Web Trouble	Unix		System Server

Module/ Component	Product	Dependency	Server	Client	Location
Switching Management	OMS Base + Switching and Schematics / DMS Base	Model Management			
Switching Service			Unix		System Server
Switching Application				Windows	Web Client
Power Flow	DMS Power Flow	OMS Base or DMS Base			
Power Flow Service		Model Management	Unix		System Server
Power Flow Applications		Web Workspace		Windows	Web Client
Suggested Switching	DMS Adv. Feeder Mgmt	Power Flow, Switching Management, and SCADA Adapter	Unix		System Server
Feeder Load Management	DMS Adv. Feeder Mgmt	Power Flow, SCADA Adapter	Unix		System Server
Fault Location, Isolation & Service Restoration	DMS FLISR	Switching Management and SCADA Adapter	Unix		System Server
Optimization	DMS Optimization	Power Flow, Switching Management, SCADA Adapter	Unix		System Server
Fault Location Analysis	DMS FLA	Power Flow, SCADA Adapter	Unix		System Server
Schematics	OMS Switching & Schematics / DMS Base	Model Management			
Schematics Generator			Unix		System Server
Generic MQ Adapters	OMS Adapters				
CIS MQ Adapter		Web Trouble	Unix		System Server
CIS MQ Callback Adapter		Web Trouble	Unix		System Server

Module/ Component	Product	Dependency	Server	Client	Location
IVR MQ Adapter		Web Trouble	Unix		System Server
Mobile MQ Adapter		Web Trouble	Unix		System Server
Generic Adapters	OMS Adapters				
AMR Adapter		Web Trouble	Unix		System Server
Storm Management	OMS Storm	Web Trouble		Windows	Web Client
Web Workspace	OMS Base DMS Base	Model Management		Windows	Web Client
Web Trouble	OMS Base	Web Workspace		Windows	Web Client
Web Call Entry	OMS Call Center	Web Trouble		Windows	Web Client
Web Callbacks	OMS Call Center	Web Trouble		Windows	Web Client
Call Overflow Adapter	OMS Call Center	Web Trouble	Unix		System Server
NMS Training Simulator	Network Management Training Simulator	Web Workspace, Switching Management		Windows	Web Client
SCADA Extensions	NMS SCADA	Web Workspace		Unix	Application Server
SCADA Adapters	NMS SCADA				
ICCP Blocks 1 & 2			Unix		ICCP Server
ICCP Block 5			Unix		ICCP Server
Generic SCADA			Unix		System Server
MultiSpeak SCADA			Unix		System Server
Service Alert	OMS Paging				
Service Alert Service		Web Trouble	Unix		System Server
Service Alert Client		Web Trouble		Windows	Web Client
NMS Schema	NMS Extractors & Schema	Model Management, Oracle GoldenGate	Unix		OUA RDBMS Server

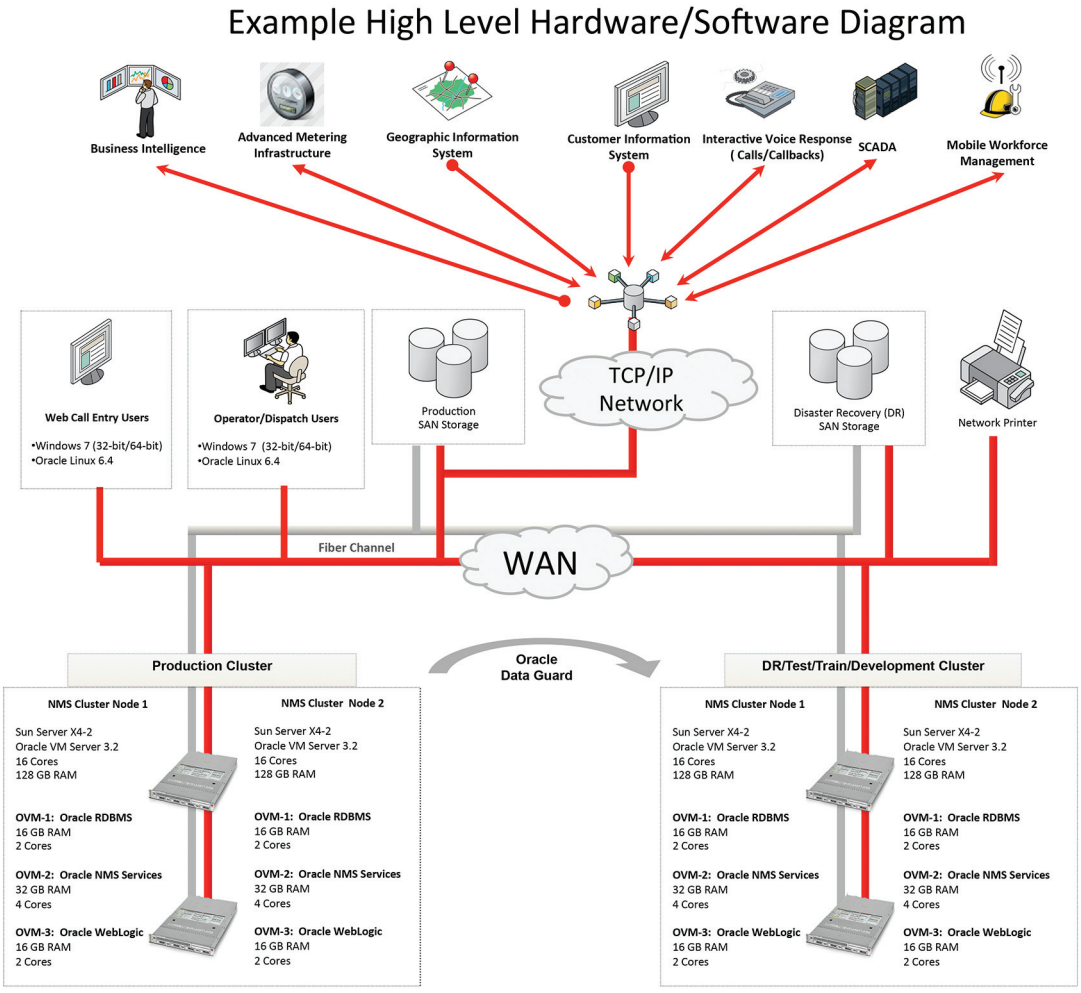
Module/ Component	Product	Dependency	Server	Client	Location
Outage Analytics	Schema	NMS Extractors and Schema, Web Trouble		Windows	OUA Web/ App Server
Distribution Analytics	Schema	NMS Extractors and Schema and Power Flow Extensions		Windows	OUA Web/ App Server

Oracle Utilities Network Management System High-Level Conceptual Diagram



Example Hardware/Software Overview

Below is an example Network Management System hardware and software overview diagram. It can be used to get a reasonable idea regarding the hardware and software a medium sized utility (500K->1M customers) might use to support an NMS installation.



Hardware Sizing

Hardware sizing guidelines are not discussed in this document. There are many variables that affect hardware sizing and the calculations would be more complex than what is suitable for this document. Hardware sizing is best handled by the Consulting Services team that is working on the project.

Chapter 2

Standard Product Implementation

This chapter provides an overview of a standard implementation of Oracle Utilities Network Management System, including:

- Overview
- Software Release Level
- Installation
- Interfaces
- Modeling and GIS Integration
- Operations Modules Software Configuration

Overview

Changes to the Oracle Utilities Network Management System standard product software, installation, interfaces, modeling, and software configuration are considered typical and acceptable for a standard product implementation. Staying within the guidelines discussed in this guide allows a customer to follow the standard configuration from release to release and significantly reduces Oracle Utilities Network Management System migration and upgrade issues.

The intent is to allow a customer to make changes that follow the 80/20 rule; that is, a customer should be able to stick to 80% of the standard product configuration and only make the 20% configuration changes which are absolutely necessary to make the implementation successful.

There are many additional configuration changes possible and technically supported by Oracle; however, changes outside of these guidelines are considered project scope changes and redefine the project as a non-standard configuration project. This in turn creates testability and maintainability issues, as non-standard configuration may not be encompassed by our test process and can result in issues with which our customer support department may not be familiar. In addition, deviations from the product configuration mean your system will not conform as closely to standard product documentation and training material.

Software Release Level

A standard product implementation should utilize a release of Oracle Utilities Network Management System with no software code changes, additions or modifications. The software should be on an officially supported release code line and not a special project code line. Only patches that are produced by the Oracle support organization and/or the project team should be installed when necessary to fix critical problems.

Installation

The installation should be done according to the guidelines taught in the Oracle Utilities Network Management System's System Administration class and follow all recommended procedures for system configuration. The software should be installed on servers and clients in a configuration that meets the requirements stated in the Architecture Guidelines section of Chapter 1 for the installed modules. The installation also should comply with the required operating system level and patches identified in the *Oracle Utilities Network Management System Quick Install Guide*. The utilized Oracle Utilities Network Management System software modules should have all dependent Oracle Utilities Network Management System software modules installed and configured. The required third-party products should be installed and at the supported release level as stated in the *Oracle Utilities Network Management System Quick Install Guide* for the installed release.

Interfaces

A standard product implementation should use the Oracle Utilities Network Management System standard CIS, IVR, and mobile data interfaces with an officially supported middleware gateway such as the WebSphere MQ gateway. SCADA system integration should be done utilizing the Oracle Utilities Network Management System Live Energy Connect ICCP Adapter, Generic SCADA Adapter, or MultiSpeak-based web services SCADA adapter. AMR/AMI and AVL integrations should be done using the Oracle Utilities Network Management System MultiSpeak Adapter. Paging and email notification integration should be done using the Service Alert module.

When interfaces are required to non-standard systems that cannot be supported by the standard interfaces described above, they should be generally implemented utilizing the published APIs and should not directly read or write to the Oracle Utilities Network Management System operations database.

Database level and/or reporting integration may be done using the Oracle Utilities Analytics database and must utilize tables and attributes described in the Oracle published schema.

Modeling and GIS Integration

The following sections describe some recommended guidelines to follow when you integrate Oracle Utilities Network Management System with a GIS.

GIS Model Extractor

The GIS extractor utilized should either be supported by Oracle or by one of our modeling partners. The extractor should produce Oracle standard model preprocessor (MP) files and utilize the Oracle conventions for model building and an approved incremental update process.

Standard Preprocessor

The Oracle Utilities Network Management System standard preprocessor supports eighteen different rules that allow for data translation (for instance, expand elbows, or add recloser bypass switch). It is acceptable to use as many of these rules as necessary to build an acceptable operations model. The standard preprocessor takes as input model preprocessor (MP) files and produces Oracle standard model build (MB) files - generally on a feeder or substation basis but possibly on a geographic tile basis.

Device Types and Attributes

Select which device types (classes) are used from the standard model definition, mapping the customer's GIS data to these existing classes.

- Define unique class alias names based on available GIS attribute(s).
- Select which attributes are used from the standard model definition (providing at a minimum those necessary for the required modules), again mapping the customer's GIS data to the existing attributes.
- Utilize Oracle-provided modeling workbooks to define the model used for the project, which is used to generate the project classes and inheritance.
- The name of any device may be constructed from one or more GIS attributes.

Software Configuration Dependencies On Device Types

There are a number of NMS software configuration aspects that depend upon the device types that are chosen to be built within the NMS data model. In so far as the data model can change for different facilities, the software configuration must be adapted. The following configuration settings are dependent upon the resulting NMS model definition and require adaptation for every project. These configurations are generated automatically by Oracle to match the defined NMS model.

- Control Tool panels
- Trouble Stop Classes
- Symbology mapping and symbol set

Operations Modules Software Configuration

This section lists configuration options in Oracle Utilities Network Management System applications and components, including:

- Web Workspace
- Web Trouble
- Web Call Entry
- Web Callbacks
- Web Switching Management
- Power Flow
- Fault Location Analysis (FLA)
- Fault Location, Isolation, and Service Restoration (FLISR)
- Feeder Load Management (FLM)
- Suggested Switching
- Optimization
- Load Shed and Restoration
- SCADA Extensions
- Service Alert
- Storm Management

Overview

Unless there is sound reason to change them, Oracle recommends that labels, buttons, table columns and dialogs be left as-is for consistency. This avoids confusion and further improves our ability to support our customers. However, there are cases where such changes are allowed, and the following sections identify those cases. There are also cases where it is allowable to delete a field, button or label. This may mean that the deleted item is actually just "hidden". Depending upon where on the form the deleted or hidden item was originally placed, there may be some "white space" remaining where the deleted item was present.

Web Workspace

Login

- Add and remove usernames (using the Configuration Assistant).
- Delete or rename user types.

Work Agenda

- Change labels of any column.
- Change labels of any menu/toolbar items.
- Add three permanent sorts.
- Change set of Work Queues (or Dispatch Groups).

Main Menus/Toolbar

- Delete or rename items.

Authority

- Define specific control zone hierarchy (up to 5 levels).

Viewer

- Change project symbology file used by Oracle Utilities Network Management System (Customer responsibility - includes AVL crew symbology if configured.
- Viewer background color may be gray or black.
- Annotation and/or landbase color may be changed to be compatible with the Viewer background. All annotation is assumed to be one color, and all landbase graphics are assumed to be a single color.
- Zoom levels.
- Declutter / reclutter.
- Big Symbols.
- Selectable and unselectable objects.

Control Tool

- Change labels for actions.
- Delete actions.

Web Trouble

Event Management Rules

- Delete any standard rule set.
- Change parameter values of any rule in any standard rule set (using the Configuration Assistant).
- Delete any rule in any standard rule set (except in cases where there are rule dependencies).

Event Details

- Delete outage reporting drop down menus.
- Rename outage reporting drop down menus.
- Add and delete items on outage reporting drop down menus (using the Configuration Assistant).
- Add additional option menu field verification prior to completion (e.g., not only must the Failure and Remedy be changed from "Unselected", but it may also check for values in other option menu fields prior to completion).
- Remove current completion validation check or any other configured validation check.

Crew Actions

- Add and Remove Crew Types from standard list of crew types.
- Add and Remove Personnel Job Titles from standard list of job titles.
- Add and Remove Vehicle/Equipment types from standard list of vehicle/equipment type.
- Add and Remove Control Zone filter interactions with the Work Agenda.

Damage Assessment

- Add, remove, or rename damage types.
- Modify the minutes to repair, and minutes to repair if inaccessible, for each damage type.
- Add, remove, or rename damage parts.

Web Call Entry

- Add and remove usernames - using the Configuration Assistant.
- Add/Remove Trouble Codes but must map to Oracle standard trouble codes .
- Change labels and order of columns in Outages Summary.
- Modify Event History Cause drop-down lists to reflect outage reporting drop down menus.
- Can be configured in standalone mode or integrated into Web Workspace. When integrated, the Outages Summary is not supported because the user should use the Work Agenda instead.

Web Callbacks

- Add and remove usernames - using the Configuration Assistant.
- Add/Remove Callback Status options but must map to Oracle standard callback statuses.
- Change labels of any column in main window and View My Callback Lists window.

Web Switching Management

Switching List/Safety List

- Change labels of columns.
- Change labels of menu/toolbar items.

Switching Documents

- Change labels on any header field.
- Delete any header field.
- Change header labels on any switching step field.
- Add additional required fields verification prior to state change.
- Remove any validation check or any other configured validation check.

Safety Documents

- Rename any safety document.
- Change labels on field of standard documents.
- Add additional required fields verification prior to state change.
- Remove any validation check or any other configured validation check.
- Delete any fields of standard documents.
- Delete any standard documents.
- Define up to three new safety documents (starting from a copy of any standard safety documents and making any of the allowable changes listed above).

Power Flow

Power Flow User Tools

- Change labels of columns on Power Flow Results.
- Remove columns to display on Power Flow Results.

Power Flow Algorithm Rules

- Change parameter values of any power flow algorithm rule - using the Configuration Assistant.

Load and Distributed Generation Profiles

- Number of day types.
- Number of load profile intervals.
- Number of Temperature Bands.
- Temperature Band Ranges.
- DER and Weather Zone Configuration.

Seasonal Conductor and Transformer Flow Ratings

- Seasonal limit.
- Normal limit.
- Emergency limit.

Power Flow Switching Extensions

- Change labels of Power Flow specific columns on switching steps.

Fault Location Analysis (FLA)

- Change labels of any column.
- Change ordering of columns.
- Change formatted string value for "Distance from Upstream Switch" column, for example from ft to yds or meters, depending on the GIS units used.

Fault Location, Isolation, and Service Restoration (FLISR)

- Change labels of any column.
- Change labels of any button.

Feeder Load Management (FLM)

- Change labels of any column.
- Change ordering of columns.

Suggested Switching

- Change labels in Suggested Switching user tools.

Optimization

- Change labels of any screen.

Load Shed and Restoration

- Change labels for any column.
- Change ordering of columns.

SCADA Extensions

- Change labels of columns on SCADA Summary page.
- Change tooltips of buttons on SCADA Work Agenda page.
- Change alarm limit values.

Service Alert

Service Alert provides a user interface for update and maintenance of the contact list, notification parameters, customer contact information, and critical/sensitive customer information; it is the customer's responsibility to do this administration via the provided tool. You may modify XSL messages for use by the supported notification mechanisms/ devices.

Storm Management

- Change labels of columns.
- Change labels of menu/toolbar items.
- Change the historical average lookup values, but not how they are used in the algorithm.
- Define a sort order for the events that is used by the analysis engine prior to stepping through its periodic analysis, within the constraints of the configuration options available for this purpose.
- Change storm outage type names, definitions and restoration order, within the constraints of the configuration options available for this purpose, including adding or removing some (but not all) outage types (directly tied to the historical average lookup value definition process).
- Specify which of the top three control zone levels is the "simulation level" (the level at which the lookup values are specified).
- Define which crew types are eligible to assess/repair which storm outage types.
- Define performance factors for each crew type.
- Define nominal crew resources.
- Change storm shift definitions, within the constraints that there must be at least one but no more than four shifts, and the sum of all shift lengths must be exactly 24 hours (directly tied to the historical average lookup value definition process).
- Change storm season definitions, within the constraints that there must be at least one but no more than four seasons, and each month must be part of a season (directly tied to the historical average lookup value definition process).
- Change storm holiday definitions, including the removal of all holiday definitions (directly tied to the historical average lookup value definition process).

- Change storm special conditions types (directly tied to the historical average lookup value definition process).
- Change storm level names, including adding or removing some (but not all) levels.
- Change list of company names for the crew resources, including adding or removing some (but not all) names.
- Add and remove usernames and passwords.
- Delete or rename user types.

Chapter 3

Isis Configuration

Isis is the backbone of the Oracle Utilities Network Management System. It is the messaging bus through which all Oracle Utilities Network Management System daemon process components communicate. This chapter provides the details for configuring Isis. It includes the following topics:

- Isis Configuration Files
- Isis Architecture
- Isis Directory Structure
- Isis Environment Variables
- Isis Log Files
- Starting Isis
- The cmd Tool
- Troubleshooting

Isis Terminology

The following table describes Isis terms used in this chapter.

Term	Definition
Ports	Isis requires a set of three TCP/IP ports for communication. These are defined in the sites file. These ports may also be defined in the /etc/services file. The port definitions in the /etc/services file may be overridden through the use of the ISISPORT and ISISREMOTE environment variables. ISISPORT defines the UDP port that Isis backbone sites use to communicate with each other and the TCP port that processes use to connect to the Isis backbone. Thus ISISPORT defines two out of the three TCP/IP ports for running Isis. Default value for ISISPORT is 2042, as registered by the Internet Assigned Numbers Authority (IANA). ISISREMOTE defines the UDP port used by processes to communicate to the Isis backbone when no TCP ports are available. Specifying this field is necessary even though it is generally not used by Oracle Utilities Network Management System. Default value for ISISREMOTE is 2043, as registered with IANA. If ISISREMOTE is NOT provided it will default to $ISISPORT+1 via the nms-isis start startup script.
Site	Each server node that runs the Isis protocol for a given $ISISPORT is an Isis backbone site. NMS only supports one site configurations, typically site = 00
protos	Protos is the name of the Isis protocol process.

Isis Architecture

The Oracle Utilities Network Management System C++ daemon services (often referred to as NMS Services) require Isis to be running on the same Unix server and username where the NMS Services execute. When Isis is running, two Isis backbone specific Unix processes will be executing: `isis` and `isis-protos` (protocol server).

Isis Directory Structure

The Isis directory structure is provided for verification purposes only.

Directory	Contents
bin	Isis executables. The cmd command resides here as well. cmd provides a command line interface to Isis that is useful for verifying connections and debugging problems.
lib	Isis runtime libraries
include	Contains Isis include files used in compiling the software

run_isis

The run_isis directory should normally be under $NMS_HOME/etc/ and contains Isis configuration files:

- sites, which defines the node for this instance of isis.
- isis.rc, which provides startup information for Isis
- isis.prm, which is the Isis parameter file. The location of the Isis parameter file can also be overridden via the ISIS_PARAMETERS environment variable.
- If not already created, the run_isis directory and its files will be created the first time you run isisboot.

Isis Configuration Files

This section addresses the files that affect the configuration of Isis software. Some of these are Isis specific files, while others are operating system files.

sites File

The Isis sites file is located in $NMS_HOME/etc/run_isis. It identifies the node on the network that will be running Isis, typically the site ID = 001 and the host IP using 127.0.0.1 (localhost), and defines the TCP/IP port numbers under which it will run.

The standard Isis sites file should always specify the standard ports you see in the example below. If you wish to override the standard Isis ports set the ISISPORT environment variable accordingly (see notes below on ISISPORT). The nms-isis start script will examine the ISISPORT environment variable and - if set to anything other than 2042 - will create an alternate sites file (sites.$ISISPORT) with the proper port configuration.

The format of this file specifies the Isis node number, network service ports, hostname (or IP address) along with an optional user name and comment. Below is a typical sites file for a single node where Isis traffic is restricted to the loopback address.

```
+ 001:2042,2042,2043 127.0.0.1
```

The leading plus sign is very important and this file cannot have any comments (except in the comment section of the end of each line). If the Isis sites file does not exist, the `nms-isis` startup script will create it, which is the recommended way to manage the sites file.

isis.rc Startup File

The isis.rc file is located in $NMS_HOME/etc/run_isis. It contains the following information:

- The Isis processes to start
- The location of Isis logs

A generic isis.rc license file is included in the Oracle Utilities Network Management Systems software distribution.

Isis Environment Variables

Isis environment variables allow Oracle Utilities Network Management System Unix processes to find the appropriate Isis messaging site.

ISISPORT and ISISREMOTE

ISISPORT is set to the second (tcp) service port in the sites file, and ISISREMOTE is set to the third (bcast) service port in the sites file. Note the second (tcp) port must be the same as the first (udp) port. These variables tell the tools and services where to listen for Isis messages. If ISISPORT is not defined, it defaults to 2042 (via the nms-isis start script). If ISISREMOTE is not defined, it defaults to $ISISPORT+1. To simplify configuration it is generally recommended to NOT specify ISISREMOTE (let nms-isis start take care of it for you).

ISIS_PARAMETERS

Specifies the Isis parameter file to be referenced by applications and services on startup.

An example Isis parameter file is provided in the templates directory (isis.prm.template). Unless there is a specific reason not to, it is suggested that the default template be used for production systems. By default (if no parameter file exists) the nms-isis start script should automatically put in place and use a copy of the default Isis parameter template file. Existing customers should verify that their Isis parameter file and the provided template are in reasonable agreement (there should be a rational explanation for differences). An example of possible parameter file content for an Oracle Utilities Network Management System appears below:

```
#isis.prm
isis_NativeThreadStackSize 131072
# specify that all applications should provide their
# parameters when a dump occurs
isis_prmDumpAllParameters 1
# allow messages which can have 10MB of information, model
# builds may require messages of this size
isis_msgMessageSizeLimit 10000000
#
isis_UDPSndbuf             131072
isis_UDPRcvbuf             131072
#
isis_iclPacketHighWaterMark 49152
isis_iclPacketLowWaterMark 32768
# don't go below 2048
isis_iclMaxSlots            4096
#PROTOS
protos_maxLocalClients 1024
protos_maxRemoteClients 1024
protos_taskHigh 100
protos_taskLow 95
```

Isis Standalone Mode

By default, Isis starts in "standalone" mode. This means that Isis will bind to the local loopback adapter (localhost - 127.0.0.1) and not the adapter defined by the gethostbyname function. This means that Isis will not be available to other hosts on the network by default.

Isis Multi-Environment Considerations

When configuring multiple Oracle Utilities Network Management System environments on a single server, each site should be assigned a unique port number (`$ISISPORT`). This logically partitions each network and prevents unwanted cross communication through a single Isis instance. As an example, the on-line system environment may be assigned to the 204x ports while the model build environment may be set to 214x, the off-line engineering environment set to 224x, and so on.

Isis Log Files

isis.[date].[time].log

The `isis.[date].[time].log` file keeps track of events while Isis is initializing and is essentially the output file for the nms-isis start script. It is located in `$NMS_LOG_DIR` (or under `$ISIS_LOG_DIR` if it is set). The `isis.[date].[time].log` file contains clues if there is any difficulty in starting Isis.

[Site No.].logdir

This is the directory where the `Protos` and `Incarnation` logs reside. The location of this directory is defined in the isis.rc file, and is typically found in `$NMS_LOG_DIR/run.isis` (or under `$ISIS_LOG_DIR/run.isis`, if `$ISIS_LOG_DIR` is defined).

The Protos Log

Protos is the Isis protocol process. This process logs its messages to `$NMS_LOG_DIR/run.isis/[Site No.].logdir.[port]/[Site No.]_protos.YYYYMMDD.HHMMSS.log` or similar under the $ISIS_LOG_DIR directory if $ISIS_LOG_DIR is defined. Check here for runtime problems with Isis. Each time Isis is restarted, the previous protos log is moved to the old_log subdirectory. The retention period for logs in the old_log directory is driven by the $NMS_DAYS_TO_LOG environment variable or for 7 days, if the environment variable is not set. Each time Isis is restarted it will delete logs older than the retention period.

The Incarn Log

There is a short file called $NMS_LOG_DIR/run.isis/[Site No.].logdir./[Site No.].incarn and it usually includes a single line containing the incarnation number for the particular site.

Starting Isis

nms-isis

nms-isis start is the script that initializes Isis.

Initializing Isis

To initialize Isis, complete these steps:

1. From the nmsadmin user name type:

   ```
   nms-isis start
   ```

   This script will return when the isis process is running. To get the status of the isis process, use this command:

   ```
   nms-isis status
   ```

   It will return "1" if the isis process is running or "0" if it is not running.

2. When complete (which could take up to a minute or more), type:

   ```
   cmd status
   ```

   This determines if Isis has successfully started and will provide information similar to the following:

   ```
   cmd:  my_site_no = 1
   my_host = 127.0.0.1
   Isis version = V3.4.14 Build: 20 $Date: 2010/06/09 19:03:03 $
   verbose mode = off
   ```

3. If it has started successfully, type:

   ```
   cmd sites
   ```

   Result: Isis lists all connected machines. For example:

   ```
   tstaix01:cesadmin$ cmd sites
   ***  viewid = 1/1
   tstaix01.oracle.com [site_no 1 site_incarn 3]
   ```

Stopping Isis

To stop isis, issue the following command:

```
nms-isis stop
```

Starting Isis on Non-Default Ports

Isis may need to run on ports other than the default ports listed in the sites file. It is common to separate different sets of services by running Isis for those services on separate network ports. For example, a configuration system may run on 1601, 1602 and 1603, while the model build services run on ports 1701, 1702 and 1702. Therefore it may be necessary to switch a client from one set of Isis ports to another.

To start Isis on a non-default port, complete these steps:

1. To check which ports to use, at the prompt type:

   ```
   echo $ISISPORT
   ```

   This returns the port configured for this environment.

2. As the nmsadmin user, set ISISPORT to the desired Isis port number. For example:

   ```
   export ISISPORT=2032
   ```

3. Then run nms-isis start as per usual. The nms-isis start script will take care of setting up Isis on the proper ports.

   Results:

   - A new sites file called sites. $ISISPORT will be created from the existing sites file.

The cmd Tool

Verify the connection to Isis using the cmd tool. If cmd is working, Isis is functioning as well. The syntax for cmd is:

```
$cmd [options]
```

Enter `cmd` from the Unix command line to bring up the command line interface, identified by the **`cmd>`** prompt. The following table presents a subset of cmd commands.

Command	Description

`sites`	Shows all nodes connected by Isis on the current ports:
	
	`cmd>sites`
	`*** viewid = 34/5`
	`test1.oracle.com [site_no 34 site_incarn 1]`
	`test2.oracle.com [site_no 33 site_incarn 1]`
	`test3.oracle.com [site_no 6 site_incarn 1]`

`status`	Provides the current status of the Isis protos process. Part of the information returned is the current Isis version corresponding to the executed cmd binary.
`list`	Provides a list of all the Isis process groups and applications connected to the protos. This can be used to identify remaining processes that are still connected to the Isis message bus.

Command	Description
`snapshot`	Sends a message to all applications currently connected to Isis to generate an Isis dump. All the Isis related information for this process is written to disk in a log file with the process ID as the prefix ([pid].log). This log file can be found in the run.*Service directories for services or the directory from which applications have been launched. Isis dumps are extremely useful when debugging problems, as they can tell the developer exactly what messages are being processed at the time the dump was generated.
`rescan`	Tells protos to update the site view.
`shutdown`	Causes the protocols process to shutdown. Wait 1 minute before restarting Isis after a shutdown or an unsuccessful start attempt, and verify that all processes are completely down by checking the process list on each node (ps -aef).
`Help`	Print all cmd command options.
`Help [command]`	Print information about a specific command.

Exiting cmd

Enter `quit` to exit cmd.

Troubleshooting

When an Oracle Utilities Network Management System application or Service is experiencing problems, some helpful information would include an Isis dump of the applications process, the log file associated with that application, and the output of the processes list.

Generating an Isis Dump File

To generate an Isis dump file, complete these steps:

1. On each node of concern:

```
ps -aef > $(hostname)_ps.out
```

2. Identify the process ID of the problem application(s).

```
grep -i [application] $(hostname)_ps.out
```

3. Use the following command to generate an Isis dump file for a specific process:

```
kill -USR2 [pid]
```

The process will not be affected and will continue to operate, but upon receiving the USR2 signal, it will generate an Isis dump (`[pid].log`) in the directory from which that tool was launched.

> **Note:** Any subsequent USR2 messages will result in the process appending a new Isis dump to the `[pid].log` file. Only the user running the applications can perform this action.

Generating an Isis Dump File for All Applications

An alternative is to issue the cmd `snapshot` command, which will create an Isis dump for all applications connected to the Isis message bus. The applications will continue to run, but every single application running will create an Isis dump file. This will clutter the file system, but it is sometimes the best way to gather all the information you need to investigate a problem.

Use the Oracle Utilities Network Management Systems nms-snapshot to execute an Isis `cmd snapshot` command and properly gather the relevant logs. Run `nms-snapshot` with no options or the `-h` option for information on what options are available.

Most commonly, `nms-snapshot` is run with the `-a` option, which will run `cmd snapshot` and get stack dumps from DBService and dbrdwr processes twice (waiting 20 seconds between snapshots/dumps), and gather all logs for the last day. It will create a zip file in `$NMS_LOG_DIR` that can be sent to Customer Support for investigation. With the full set of logs, Customer Support can track interactive messaging for problem investigation and resolution.

Reporting a Problem to Customer Support

In general, when reporting a problem to Customer Support, the following information can speed the problem identification and resolution process:

- An explanation of what the observed symptoms were and where they occurred.
- An explanation of how to repeat the problem, if possible.
- An explanation of expected behavior.
- A specific time frame when the problem occurred.
- Example data demonstrating the problem (e.g., event numbers, crew names, etc.).
- Service logs and environment log files of the affected Services/Applications.
- Isis dumps of the affected application and services at the time the problem was observed. A complete Isis dump of all processes may be requested if the problem is repeatable, along with a process list output file.
- The core file trace, if a core file exists for the process.
- Any other activity that occurred prior to, or concurrent with, the issue that may stand out as a possible contributor.

Chapter 4

Information Lifecycle Management

Information Lifecycle Management (ILM) is a concept - not a technology. The primary concept behind ILM is that as the need to access data decreases it can generally be moved to more economical storage and possibly (ultimately) deleted. Implementing an ILM strategy for a given data set can help manage storage costs and possibly improve performance. The core technology on which ILM is implemented is Oracle RDBMS partitioning.

Prerequisite

To implement the standard NMS ILM strategy requires the Oracle RDBMS Partitioning option. Oracle Partitioning is a separately licensed module that also requires Oracle RDBMS Enterprise Edition.

NMS provides ILM support for the NMS model management RDBMS tables. NMS model managed tables always include birth and death DATE columns. The death column tracks when a row was taken OUT of the active model. The death column is NULL for active records.

Product support for ILM requires that the NMS model be initially created with ILM (partitioning) turned on. The NMS product includes Oracle RDBMS Data Definition Language (DDL) scripts that are generally used to create the core NMS model managed tables. For example, the `$NMS_SQL_FILES/nms_model_ops.sql` file contains DDL statements for most core NMS model tables.

Note for model migration: You cannot directly alter an existing (non-partitioned) table into a partitioned table. A table is either partitioned or not when it is created. It should be possible to define a new (empty) partitioned table and "swap" the contents with an existing (non-partitioned) table. Essentially this amounts to copying old table data into a new partitioned table – then dropping the old table and renaming the new one to match the old. The Oracle DBMS_REDEFINITION mechanism may help with this process but the details are left as a project exercise.

If the `$NMS_DATA_FILES/[project]_licensed_products.dat` file has the "partitioning" option uncommented then, when the NMS model is being initially created (using `nms-setup` with the `-clean` option), NMS model managed tables will be partitioned. The NMS model workbook references the same "partitioning" option and should correctly create proper DDL statements (partitioned or otherwise) for the potentially project specific NMS model managed attribute tables.

The NMS ILM partition scheme uses "PARTITION BY RANGE" with the "INTERVAL" option set to 1 month and the "ENABLE ROW MOVEMENT" option set for each NMS model managed table. This will automatically create a new NMS model managed table partition each time the death column is set to a new non-NULL month date value.

The "PARTITION BY RANGE" option is applied to the virtual `ptn_date` DATE column that must exist in each NMS model managed table that we need/want an ILM scheme for. The `ptn_date` column is required to support NMS ILM partitioning but can be safely ignored for non-partitioned NMS models. When the death date is set, generally via the standard NMS incremental model build process, the row is deactivated and moved into the appropriate partition (by month of deactivation).

When NMS ILM is enabled the active NMS model exists entirely in the default `mdl_alive` partition within each of the various NMS model managed tables. This happens by virtue of the virtual `ptn_date` column being set to '1776-07-01' when the death column is null (the row is still active). It is necessary to use the virtual ptn_date column for partitioning because partitioning by date requires the partition by date column always be non-null.

As the NMS incremental model build process deactivates rows in the various NMS model managed tables new partitions will accumulate, containing the deactivated NMS model managed table rows. It is up to the project to drop unneeded partitions once they are deemed no longer relevant. To see the partitions within a given schema, the following query may be useful:

```
select PARTITION_NAME, TABLE_NAME, HIGH_VALUE from
user_tab_partitions;
```

To drop a partition from a table, use a SQL statement similar to the following - where SYS_P522 is an Oracle RDBMS generated partition for the ALIAS_MAPPING table:

```
alter table ALIAS_MAPPING drop partition SYS_P522;
```

History Tables

Oracle Utilities Network Management System includes various tables for recording and trending historical performance of SCADA analogs, feeder loading, and device violations. The data for this historical performance are stored in the description below. As with the model management table, it is strongly recommended that Oracle RDBMS partitioning technology be used to manage the contents of these tables in a more economical fashion, and, ultimately, allow retrieval (and eventual deletion) in an efficient manner. The nms-setup script creates these tables as partitioned if the partitioning option is active in the $NMS_DATA_FILES/[project]_licensed_products.dat file. Similarly, if a migration from a previous version is performed, the tables are created as partitioned by the migration scripts.

As with the model tables, the partitioning scheme used is to PARTITION BY RANGE on a date column. The tables that are partitioned and their range column are:

Table Name	Range Column	Contents
SCADA_ANALOG_HISTORY	SCADA_TIME	History of SCADA analog values.
FLM_FDR_LOAD_HISTORY	ANALYSIS_DATE	History of Feeders' FLM performance.
FLM_FDR_LOAD_DETAILS_HISTORY	ANALYSIS_DATE	History of devices' FLM performance.
FLM_DEV_VIOLATIONS_HISTORY	UPDATE_DATE	History of the devices' violations.

If the tables already exist in the schema as non-partitioned, then migration scripts will only alter them to the extent that modified columns are added, updated, or deleted. It is not possible to directly modify a non-partitioned table into a partitioned table; to do so requires a manual migration process.

If there is no historical data in FLM_FDR_LOAD_HISTORY, FLM_FDR_LOAD_DETAILS_HISTORY, FLM_DEV_VIOLATIONS_HISTORY, or SCADA_ANALOG_HISTORY, then it might be worthwhile to configure NMS for partitioning, dropping the aforementioned tables and then running the migration. The migration will create the tables (with partitioning) if they are not present.

If there is historical data in those tables that is worth keeping, then they will need to be semi-manually migrated. To do that, the following steps should be performed after partitioning in NMS is activated:

1. Run the migration script to add the new columns to the existing tables.
2. Rename the tables with new names (for example, FLM_FDR_LOAD_HISTORY_0000).
3. Re-run the migration script (this will have to be done manually with `ISQL -admin` and not `nms-apply-migrations`) to create new versions of those tables with partitioning.
4. Copy the data from the original tables to the new ones with commands like:

```
insert into flm_fdr_load_history ( select * from
flm_fdr_load_history_0000);
```

5. Run `nms-setup` to re-create triggers and procedures.
6. Drop or archive the copy tables.

The large quantity of data requires a Lifecycle Management process to be in place to prevent unbounded growth of the tables. Product configuration scripts create an RDBMS stored procedure (PURGE_HISTORY_TABLES) that will prune these history tables.

> **Note:** The RETAIN_HISTORY_RECORDS entry in CES_PARAMETERS configures how many weeks of history to preserve on-line. By default, the value is four weeks.

Oracle recommends that you create a cron table entry that will run the PURGE_HISTORY_TABLES procedure weekly (for example, 1:00 a.m. on Sunday) using the following DDService command:

```
Action any.DDService purge_history
```

To purge all of the existing Feeder Contingency Analysis results, use the following PFService command:

```
Action any.PFService purge_fca
```

To purge all of the existing Feeder Data Analysis results, use the following PFService command:

```
Action any.PFService purge_fda
```

Chapter 5

Database Configuration

Oracle Utilities Network Management System currently supports the Oracle Relational Database Management System (RDBMS). The RDBMS must be properly installed and configured prior to using the Oracle Utilities Network Management System software. This chapter provides the configuration requirements for Oracle. It includes the following topics:

- Oracle Installation Guidelines
- Oracle Tablespaces
- Oracle Users
- Starting Oracle

Oracle Installation Guidelines

It is recommended that Oracle Enterprise Edition be installed. Please see the Oracle RDBMS installation documentation for specific Oracle installation requirements.

Oracle Tablespaces

Every Oracle Utilities Network Management System must have its own Oracle tablespace set. In general, the tablespaces consist of the following:

Tablespace	Description
Production	The production tablespace (nms_db) contains all of the production data for Oracle Utilities Network Management System. This includes model data, outages, and data that is produced by operations performed in Oracle Utilities Network Management System.
Production Temporary (Optional)	The production temporary tablespace (nms_tmp) temporarily stores operating data prior to insertion into the production tablespace. The default Oracle TEMP tablespace should be the designated temporary tablespace for the system. Oracle is more efficient when managing temporary data in this way. Make sure that a sufficient amount of space is allotted to TEMP.

Tablespace	Description
Production Index	The production index tablespace (nms_idx) contains all of the indexes for the production tablespace. The nms user's .nmsrc file must contain the NMS_INDEX_TABLESPACE environment variable referencing this tablespace. Note that on a modern RDBMS (with solid state storage), it should generally not be necessary to set this variable.
Customer Data	The customer data tablespace (`[project]_customers_db`) belongs to the customer. It is populated with the entire customer database by the CIS extraction process. Public synonyms are assigned to the customer tables and selectability is granted to production Oracle users so that the necessary table joins can be created.

Each tablespace should be located on a separate disk to enhance performance and decrease bottlenecks due to high volumes of input/output. Note that if the RDBMS uses solid state storage this is generally irrelevant and can be ignored.

It is key that the tablespaces are provided with sufficient disk space and are monitored regularly for growth. When a tablespace runs out of disk space, operational data will be lost and Oracle Utilities Network Management System services will discontinue to function properly.

Oracle Instances

For performance, scalability and simplicity there is normally only one Oracle instance on a production machine. It is not generally recommended that a production machine have multiple Oracle instances on the same machine. An exception would be where a cluster is used; you may want an Oracle instance installed on both sides of the cluster (production on the primary side, Model Build, Test, or Oracle Utilities Analytics on the secondary side). Under normal circumstances there would only be one instance of Oracle on each side – if one side of the cluster fails you could end up with two instances on the surviving node. In general, try to keep it simple.

You should consult with your Oracle Utilities Network Management System Professional Services technical team to develop an appropriate solution to meet your specific needs.

Oracle Utilities Network Management System uses an Oracle Wallet and the environment variables RDBMS_ADMIN and RDBMS_HOST to create a connection to the Oracle database. The wallet stores the database users and passwords in encrypted form. RDBMS_ADMIN and RDBMS_HOST are different tnsnames aliases for the same database, each of which points the Oracle client software to the correct user/password pair within the wallet (RDBMS_ADMIN for the admin user and RDBMS_HOST for the read/write user). The wallet and related environment variables are created and maintained by running the script nms-env-config as the NMS administrative user.

Each instance of Oracle Utilities Network Management Systems (for example, production, test, model build) must have a unique database owner/schema with its own tablespaces. Using the same database owner/schema for two implementations will result in corrupted data.

> **Note:** Two or more Oracle Utilities Network Management System instances on a single machine can be acceptable (depending on machine resources) for testing, training and model build environments.

It may be necessary to tune the Oracle RDBMS for the specific environment it will be operating on. Typically a qualified DBA can perform the necessary tuning and modifications. Often this is an iterative process that requires running the full Oracle Utilities

Network Management System on the production machines and capturing statistics for analysis.

Other Environment Variables

Other Oracle-specific environment variables may need to be different between systems, but these are due to how the DBA has constructed the environments. Other than the NLS specific environment variables noted below, these are listed in one of the example tables in chapter five.

When Oracle is loaded onto a given platform, the Oracle instance itself will generally have a default National Language Support (NLS) setting. Oracle Utilities Network Management System client applications (like DBService) which utilize the Oracle Call Interface (OCI) need to know what NLS settings to use for inserting and interpreting result sets from Oracle. Presently, the easiest way to do this is as follows:

1. Add the following environment variable to your .nmsrc file: NLS_LANG

   Note: For a US configuration, Oracle believes the NLS_LANG environment variable (as far as OCI is concerned) typically defaults to AMERICAN_AMERICA.AL32UTF8. Thus if a customer sets their Oracle NLS to something other than this value (inside of Oracle during instance setup) - and does not specify the NLS_LANG environment variable to appropriately match, DBService will not start. You will see a note in the DBService log file indicating a mismatch that must be rectified.

2. The following process should work for setting NLS_LANG:

   ```
   Set NLS_LANG to
   [NLS_DATE_LANGUAGE]_[NLS_TERRITORY].[NLS_CHARACTERSET]
   ```

   where each "NLS component" needs to match the values returned by this query:

   ```
   SELECT * FROM v$nls_parameters WHERE parameter IN
   ( 'NLS_DATE_LANGUAGE',
   'NLS_CHARACTERSET', 'NLS_TERRITORY')
   ```

   For example, we have NLS_LANG=AMERICAN_AMERICA.AL32UTF8, and our query returns:

   ```
   PARAMETER             VALUE
   ----------------------------
   NLS_DATE_LANGUAGE    AMERICAN
   NLS_TERRITORY        AMERICA
   NLS_CHARACTERSET     AL32UTF8
   ```

Oracle Users

Once the tablespace is established, you must create users and grant their permissions. Oracle users are those users that have access to the Oracle tablespaces. Before defining the users, it is important to discuss the security role that a user can possess.

Security Roles

Security roles determine the level of database operations that a user can perform. There are two types of security roles:

Role	Description
ces_rw	Read-write role. This role has read and write privileges to the production data. It can delete from, update to, and insert to, all of the production tablespace objects.
ces_ro	Read-only role. This role can only connect and select data from the production tablespace objects. **Note:** Certain security tables, such as ces_users, are excluded from the view of the ces_ro role.

Users

There are three Oracle RDBMS schema/users for each Oracle Utilities Network Management System instance. Each user directly relates to the tablespaces. Substitute specific customer name for [project] where noted below.

User	Description
[project_admin]	The [project_admin] Oracle user is the owner of the admin schema and the production tablespace. This user has a ces_rw role and maintains full Data Definition Language control of the data elements in the production tablespace. This user is used during initial setup and when installing patches.
[project]_1	The [project]_1 Oracle user is the owner of the read/write schema. The user has the ces_rw role. This schema only contains synonyms to objects in the [project]_admin schema and has select/insert/update/delete/execute (Data Manipulation Language) permission on [project]_admin schema objects. This user is used for most day-to-day operations.
[project]_1_ro	The [project]_1_ro Oracle user is the application user. This user has a ces_ro role to the production tablespace.
[project]_2	The [project]_2 Oracle user has the same purpose as the [project]_1 user, except it is used by the second environment in dual-environment configuration.
[project]_2_ro	The [project]_2_ro Oracle user has the same purpose as the [project]_1_ro user, except it is used by the second environment in dual-environment configuration.
[project]_customers	The [project]_customers user has full privileges to the customer data tablespace only and no privileges on the production tablespace.

Starting Oracle

Complete the following steps to start Oracle:

1. Login as oracle. If logged in as the root user, the system will not request a password. At the prompt, enter:

   ```
   su - oracle
   ```

2. Login to SQL*Plus:

   - As the oracle user, enter:

     ```
     sqlplus /nolog
     ```

   - At the SQL> prompt, enter:

     ```
     connect / as sysdba
     startup
     quit
     ```

3. Start the listener.

4. As the oracle user, enter:

   ```
   lsnrctl start
   ```

   Note: The tnsnames.ora and listener.ora files must be properly configured to start the oracle listener. The location of these files may vary by system, but they must be consistent on all machines requiring connections via SQLNET.

5. Login as the distribution user and test the connection to Oracle. At the prompt, enter:

   ```
   ISQL
   ```

This references the RDBMS_USER, RDBMS_PASSWD and RDBMS_HOST to establish the connection to the database. If this connection is successful, a SQL> prompt will appear.

Chapter 6

Environment Configuration

This chapter includes the following topics:

- CES_PARAMETERS
- Encrypting Configuration Parameters
- The System Resource File
- Modifying Environment Variables

CES_PARAMETERS

The CES_PARAMETERS table stores various configuration options used by NMS and some of its adapters.

CES_PARAMETERS Schema

Column Name	Column Type	Description
APP	VARCHAR2(32)	Application name
ATTRIB	VARCHAR2(64)	Name of parameter
VALUE	VARCHAR2(255)	Value of parameter
MODIFIER	INTEGER	Additional modifier (optional)
SITE	VARCHAR2(255)	The name of the site this parameter applies to or "all" if it applies to all sites. Default: "all"

The APP column is "NMS" for NMS configuration parameters. Other values are used for specific adapters.

Site-specific Parameters

Some parameters are site-specific (see the documentation for each individual parameter). For these parameters, the SITE column must be the NMS_SITE_NAME environment variable at the site to which the parameter applies. For each such parameter, one entry must exist for each site.

Parameters which are not site-specific should have a SITE value of `all`.

Environment-Specific Parameters

In Dual-Environment Configuration, some parameters are environment-specific (see the documentation for each individual parameter). Different values are used in environment 1 and environment 2. These parameters are configured in two different files as described below

Configuration

CES_PARAMETERS entries are configured in the following three files.

[project]_parameters.sql

Parameters which are neither site-specific nor environment-specific are configured in this file.

[project]_site_parameters_1.sql

Parameters which are site-specific or environment-specific are configured in this file. This file is used for environment number 1 based in the NMS_ENVIRONMENT variable.

[project]_site_parameters_2.sql

Parameters which are site-specific or environment-specific are configured in this file. This file is used for environment number 2 based in the NMS_ENVIRONMENT variable.

NMS_PARAMETERS_VIEW

The NMS_PARAMETERS_VIEW view in the database is a view of the CES_PARAMETERS table that only includes the parameters that apply to the current site. This view includes all columns in CES_PARAMETERS except for SITE.

Encrypting Configuration Parameters

Some environment and configuration parameters contain sensitive information, such as authentication credentials, that should be protected. This section provides two methods for encrypting cleartext strings.

Encrypting Passwords with Oracle WebLogic Server Utility

Passwords stored in the ces_parameters table and the CentricityServer.properties, and AMRInterface.properties files can be stored in encrypted form. These passwords are encrypted/decrypted using an encryption key that is unique to each WebLogic domain where the NMS application is deployed.

> **Note:** This also applies to the BI Publisher password that is stored in CES_PARAMETERS. Even though BI Publisher runs in a separate domain, the password encryption needs to be done against the NMS domain.

To generate an encrypted password, run the following commands as the WebLogic user:

```
$ cd [domain-dir]/bin
$ . ./setDomainEnv.sh
$ java weblogic.security.Encrypt
```

This will prompt for the password and then output the encrypted version of that password, which can then be copied to the appropriate SQL or properties file.

The System Resource File

The System Resource file (`$HOME/.nmsrc`) houses the environment variables that enable the Oracle Utilities Network Management System to operate correctly and consistently. They define the connections information for the database and Isis, as well as environment specific configuration settings, such as viewer symbology.

You will need to modify the System Resource file in part for application to specific systems. One suggestion is to use environment variable dependencies. By doing this you can simplify the process of changing values; by changing one variable that is a root dependency, the change will cascade through a number of others, limiting your required changes and maintaining consistency throughout the file.

Modifying Environment Variables

Because of the innate flexibility allowed by environment variables, there are an infinite number of permutations you can apply for a system setup. Not everything you can do with these variables should be done. This section describes the suggested settings that you should adhere to in order to avoid confusion.

To modify the environment variables, complete these steps:

- Modify the variable you want to change with the new settings in the `.nmsrc` file by running the `nms-env-config` script (see below).
    - Enter "`source .nmsrc`" at the prompt to source the file in the current working environment. The new variables replace the old variables.

    Note: New variables replace the old variables when the file is sourced. You should source `.nmsrc` each time you change the file. The file `.profile` automatically sources `.nmsrc` at startup.

nms-env-config

The `nms-env-config` script is used to modify environment variables in the `.nmsrc` file, as well as the two wallets that are used to store passwords for the database and connection to the application server. When run with no flags, `nms-env-config` will prompt you to set all aspects of environment variables and wallets. However, there are some command line flags you can add to shorten the task. For example, if you only want to update the database wallet, add "`--update-db-wallet`". The list of arguments is as follows:

```
-h, --help            show this help message and exit
--db-config, -d       Configure database variables and wallet.
--base-config, -b     Configure NMS base environment variables.
--appserver-config    Configure variables and wallet for the
                      application server.
--adapter-config      Configure MQ, IVR, and RDBMS/File-based
                      SCADA variables.
--translation-config  Configure translation variables.
--update-db-wallet    Update the database wallet password.
--update-nms-wallet   Update the NMS (app server) wallet password.
```

Environment Variables

The table below lists the required environment variables and their standard settings that must be modified depending on the type and number of environments you are constructing. See templates/nmsrc.template for more variables. Other variables may be added as well, depending on the functionality of your system. This is not an exhaustive list, but it does address the variables typically required to start an Oracle Utilities Network Management System.

Environment Variable	Example Setting	Description
DATEMSK	`$NMS_HOME/etc/nms_datefmt` • In dual-environment configuration, this must be set to a different value in each environment.	This file will be generated and updated by Oracle. It defines all the expected date formats that can be encountered as input by widgets and Services. Services will use the values in this file, for example, as a format dictionary when given call time as part of a trouble call. Expected time formats should be placed near the top of the file so that the search and compare algorithm encounters the most likely values as quickly as possible.
ISIS_PARAMETERS	`$NMS_HOME/etc/run_isis/isis.prm`	Identifies which file to reference for Isis parameters. This must be established before initiating an application.
ISISPORT	System specific, the default should be 2042.	A TCP/IP connection port on which Oracle Utilities Network Management System processes communicate (via Isis).
ISISREMOTE	System specific, the default should be 2043.	A TCP/IP port used when you are making a connection to a "remote" protos. This can either be when the process is running on a machine without protos or if a local connection is attempted and all the local connections are filled. In dual-environment configuration, this must be set to a different value in each environment.
NLS_LANG	System specific. Example: `AMERICAN_AMERICA.WE8MSWIN1252`	The National Language Support value for the Oracle database installation. DBService will not start unless this is set correctly. To definitively determine what the various NLS_LANG components should be for your RDBMS instance, the following query should be helpful: select * from v$nls_parameters where parameter in ('NLS_DATE_LANGUAGE', 'NLS_TERRITORY', 'NLS_CHARACTERSET')
NMS_APPSERVER_HOST	System specific. Example: `server.example.com`	The hostname of the WebLogic Java application server.
NMS_APPSERVER_PORT	System specific: Example: `7001`	The port on which the Java application server at NMS_APPSERVER_HOST is listening. The WebLogic default port is 7001. In dual-environment configuration, this must be set to a different value in each environment.

Environment Variable	Example Setting	Description
NMS_APPSERVER_PROTOCOL	`https`	The protocol used to access the Java application server. Valid values are https and http. https is the default and highly recommended.
NMS_BASE	`$NMS_ROOT/nms/product/2.5.0.0`	This environment variable is set to the product installation directory for the active installation.
NMS_CONFIG	`$NMS_HOME/`*`[project]`*	This is the location of the project configuration and implementation files. The project name (for example, OPAL) must also match the `NMS_CONFIG_ORDER` variable on the left side (for example, "`OPAL nms`").
NMS_CONFIG_ORDER	Project specific. Example: `[project] nms`	Defines the configuration inheritance path for a system. When the setup process executes, it searches this site variable from left to right looking for configuration files with prefixes that match the value in the site variable. This feature lets you inherit or override the nms configuration. This variable is used by most of the configuration scripts.
NMS_DATA_FILES	`$NMS_HOME/sql`	Defines the location of data files used in various scripts and routines that define aspects of system configuration. This variable must be defined and can be accessed from a number of scripts. The standard location for these files is the `$NMS_HOME/sql` directory. Examples arenms_classes.dat, nms_inheritance.dat, and OPAL_devices.cel.
NMS_DATA_TABLESPACE	`ces_db`	Used by the `ISQL` process to identify the tablespace name of data tablespace.
NMS_DAYS_TO_LOG	`5`	Identifies how long to store the old log files. When services are restarted, log files older than the set number of days (5 in this case) will be removed.
NMS_DOMAIN_SUFFIX	`example.com`	The domain suffix to be used when sending emails via Service Alert.
NMS_ENVIRONMENT	`1`	Indicates which NMS environment this is. This must have a value of either "1" or "2." In dual-environment configuration, this must be set to a different value in each environment.
NMS_HOME	`${HOME}`	The **nmsadmin** username home directory. This is the directory where the implementation directory and runtime directories exist. This should be set to the **nmsadmin** username home directory.

Environment Variable	Example Setting	Description
NMS_INDEX_TABLESPACE	`ces_idx`	Used by the `ISQL` process. It will parse SQL scripts that create indexes and make sure that the index is actually created in the specified tablespace name. This tablespace must be owned by the user configured in the Oracle Wallet. The practice of separating indexes from operational data may improve Oracle performance, but is often not necessary on an Oracle RDBMS installation that leverages solid state storage or general purpose SAN storage.
NMS_CLOB_SZLIMIT	15	Used by DBService processes. Sets the size limit (in MB) for fetching CLOB columns from the database. If column value is larger than the specified limit than truncated value will be returned. Range of valid values is 1 through 100 MB. Default value is 10MB.
NMS_LOG_DIR	`$HOME/log`	For services and login environments, this defines where to place the resulting log files. Since log file generation requires write access for a process, the user who started the process must have write access to this directory. It is highly recommended that this directory be located on a different file system than `$NMS_BASE`. Writing NMS logs to a separate file system reduces the likelihood of logs filling up a file system, which could impact the execution of NMS Service processes.
NMS_MODEL_SOURCE	System specific. Example: `/users/nmsadmin2/data`	In a dual-environment configuration, this must point to the $OPERATIONS_MODELS directory of the other environment. Otherwise, this variable must be blank. This is used for syncing $OPERATIONS_MODELS when switching environments.
NMS_NS_HOST	System specific. Example: `server.example.com`	The hostname where the Naming_Service is started. Only needed for sites running a Java application server.
NMS_NS_PORT	System specific. Example: `17821`	The port on which the Naming Service is running. Only needed for sites running a Java application server. In dual-environment configuration, this must be set to a different value in each environment.
NMS_ROOT	`/users/nmsadmin`	Provides a common location to place the base Oracle Utilities Network Management System directories and files owned by the administrator. It is recommended that you set this to the home directory of the Oracle Utilities Network Management System administrator. By specifying this directory correctly, you can use it to simplify other installations. When this value changes, the change will be cascaded throughout the other dependent environment variables. This environment variable is used by a number of scripts and processes.

Environment Variable	Example Setting	Description

NMS_SITE_NAME	`System specific`
	
	Example:
	`nms-a.example.com`
		This represents the name of the site the local host belongs to. The nms-env-config script defaults this to the FQDN where nms-env-config is run if no value is entered by the user, but any name can be chosen up to a maximum length of 255 characters. This value must be unique between NMS hosts and identical on both environments of a host when Dual-Environment Configuration is used.

NMS_SMTP_SERVER	`smtp.example.com`	The hostname or IP address of a Simple Mail Transfer Protocol (SMTP) server. This is used by Service Alert when sending alert emails. See also: NMS_DOMAIN_SUFFIX.
NMS_SQL_FILES	`$NMS_HOME/sql`	Defines the location of SQL files used by NMS. This variable must be defined and can be accessed from a number of scripts. The standard location for these files is the $NMS_HOME/sql directory.
NMS_SSL_CERT	`$NMS_HOME/etc/` `nms_cilent_cert.pem`	Location of the client SSL certificate which is used by SwService to establish secure connection to the WebLogic server. Security chapter covers the process for creating such certificate.
NMS_SYSDATE	Environment specific. Example: `%D%R|%D|%R`	Defines the display format for which all applications will display date and time elements. The format for this requires specifying 3 formats: date and time | date | time The three formats specified in this environment variable must also be added to the $DATEMSK file.
OPERATIONS_MODELS		Directory containing model files. Traditionally set to `${NMS_HOME}/data`.

ORACLE_HOME	System specific:
	
	Example:
	`/usr/users/oracle/`
	`product/11`
		Identifies the home directory for the Oracle user. This is necessary to simplify other variables dependent on this path.

ORACLE_READ_ONLY_USER	System specific. Example: `nms_1_ro`	The read-only database log in name.
ORACLE_READ_WRITE_USER	System specific. Example: `nms_1`	The read-write database log in name.

Environment Variable	Example Setting	Description
ORACLE_SERVICE_NAME	System specific	The service name of the Oracle database that the system should connect to.
ORACLE_SID	System specific. Example: `PRODSERV01`	Identifies the Oracle session ID value.
PREFERRED_ALIAS	Model specific. Example: `OPS:PSU`	Defines what alias of a device is to be displayed by default. In the example, the system will display the alias that has a DB_TYPE of OPS as found in the alias_mapping table. If an alias with a DB_TYPE of OPS does not exist, then the PSU (pseudo) alias will be displayed. This, by convention, is a unique name of [class_name.device_idx]. Depending on the model build definition, you can use and define other alias options, such as a SCADA alias.
RDBMS_ADMIN	System specific. Example: `PRODSERV01ADM.world`	The tnsname for establishing a sqlnet connection via Oracle to the admin schema. This value must exist in the tnsnames.ora file on the system attempting a connection. This is required for the use of many setup scripts and `ISQL -admin`.This value must be different from RDBMS_HOST.
RDBMS_HOST	System specific. Example: `PRODSERV01.world`	The tnsname for establishing a sqlnet connection via Oracle to the read/write schema. This value must exist in the tnsnames.ora file on the system attempting a connection. This is required for the use of many setup scripts and ISQL. This value must be different from RDBMS_ADMIN.
SYMBOLOGY_SET	System specific. Example: `$OPERATIONS_MODELS/SYMBOLS/PRODUCT_SYMBOLS.sym`	Identifies the primary symbology file loaded by the Viewer. This file identifies the Viewer symbols for objects that do not have Scalable Vector Graphic (SVG) symbols.

Dual-Environment Configuration

Oracle Utilities Network Management System has the option to use a dual-environment configuration to reduce downtime when installing patches.

Dual-environment configuration requires:

- Two Unix users that are members of the same Unix group. Each user has their own installation of NMS executables.
- Two WebLogic Managed Servers on the same WebLogic node. Each managed server must use a different port.

In this configuration, each environment is numbered 1 or 2 (based on the NMS_ENVIRONMENT environment variable). Only one of these environments can be active at a time.

While one environment is active, most of the steps for installing a patch can be performed on the other environment without interrupting users logged in to the active environment.

The following environment variables must have different values in the separate environments:

- ISISPORT
- ISISREMOTE
- NMS_APPSERVER_PORT
- NMS_ENVIRONMENT
- NMS_MODEL_SOURCE
- NMS_NS_PORT
- ORACLE_READ_ONLY_USER
- ORACLE_READ_WRITE_USER

The following parameters in the CES_PARAMETERS table must have different values in the separate environments:

- WEB_corbaInitRef
- WEB_mapDirectory
- WEB_mapHttpdPort
- WEB_oemHttpdPort
- WEB_tempDirectory

The `$NMS_DATA_FILES/model-sync-filter.cfg` must exist and be a valid rsync filter configuration file. This is used by the `nms-sync-model` script (run by `nms-setup` and `nms-post-setup`) when syncing $OPERATIONS_MODELS from the other environment to the current environment. `$NMS_BASE/templates/model-sync-filter.cfg.template` is a template that can be used to create this file. Run the following command to copy the template:

```
cp $NMS_BASE/templates/model-sync-filter.cfg.template $NMS_CONFIG/sql/
model-sync-filter.cfg
```

Edit if necessary to exclude any files or directories that should not be copied between environments, such as temporary files. In most cases, the template can be used as-is. Note that there should not be any blank lines in the file. Running `nms-install-config` will install the file from `$NMS_CONFIG/sql` to `$NMS_DATA_FILES`. More information about installing patches in this configuration can be found under Steps to Deploy a Patch Bundle in Dual-Environment Configuration in the *Oracle Utilities Network Management System Installation Guide*.

Chapter 7

Services Configuration

The configuration of Oracle Utilities Network Management System services involves establishing the location of system services on server nodes in the computer network and defining their configuration and command line options.

This chapter includes the following topics:

- Services Overview
- Service Alert Email Administration
- Service Alert Script Configuration
- Services Configuration File
- Model Build System Data File
- Starting and Stopping Services

Services Overview

Oracle Utilities Network Management System services provide memory-based model management for RDBMS persistent electrical network model information - generally to support real-time access and performance objectives. The services maintain the memory resident data model for the real-time status of the electrical network. The memory model caches the necessary data to build the model from relational database tables. The services then solve this model (fills in the blanks, determine what is energized, grounded, looped, etc.) and optimize the result for client access. Each service generates and passes appropriate incremental model updates to Isis (the Network Management System real-time publish/subscribe message bus) for publication. Interested applications subscribe to the published messages to keep the Network Management System end users up to date with current state of the model.

Startup scripts that run when the operating system boots can be used to automatically start the Oracle RDBMS, Isis, and Oracle Utilities Network Management System services. How you configure and where you place these scripts is based upon startup (default) Unix "runlevel" and your platform. For Linux platforms you can generally determine your current runlevel via:

```
/sbin/runlevel
```

For Linux, startup/shutdown scripts are generally located in the `/etc/init.d` directory. A Unix softlink to each startup script to run for a given runlevel is generally made in the `/etc/rc<run_level>.d` directory. Other Unix operating systems have similar but often slightly different conventions. It is presently an exercise left to the system administrator to

properly create and configure startup scripts that will properly run on startup for a given Operating System.

Oracle Utilities Network Management System Services are generally flexible and attempt to cater to the functional needs of various utility clients through the use of command line options and run-time parameters stored in the relational database. Below is a brief summary of the primary Oracle Utilities Network Management System Services. Details about available command line options and relational database parameters specific to each service can be found in Appendix A - Command Line Options and High Level Messages.

SMService - System Monitor Service

SMService monitors the core Oracle Utilities Network Management System service and interface processes. It reads and caches the appropriate system.dat configuration file to determine which processes to initiate and monitor. In the event that a managed process fails, SMService restarts it based on the cached configuration data from the system.dat file.

The following variations of system.dat files should be located under $NMS_HOME/etc. There should be *.template versions of these files in the `$NMS_BASE/templates` directory. These configuration files generally define the specific run-time executables and command line options necessary for a given Network Management System installation:

- system.dat.init - defines configuration required for initial setup.
- system.dat.model_build - defines minimum configuration required for initial model builds.
- system.dat file - defines configuration for fully operational Network Management System.

sms-start will launch SMService, which in turn will cache the `$NMS_HOME/etc/system.dat` file by default and then launch the remaining services, interfaces, and adapters as defined by the `$NMS_HOME/etc/system.dat` file. The following command sequence can be used to specify an alternate system.dat type file:

```
sms-start -f ~/etc/my_system.dat
```

When the sms-start script is called without command line options, it instructs SMService to validate that all processes it is configured to manage are in fact running. The "configuration" that is used by SMService for this validation is from the previous "sms-start -f system.dat" type request. If you make changes to your "system.dat" type configuration file, you can use "sms-start -f system.dat" to get a running SMService to "adopt" the updated configuration (no need to restart SMService). You can use this technique to change command line options for SMService managed processes but be aware that SMService will NOT automatically stop/restart processes. If a process is no longer managed by SMService after configuration, you will need to manually stop that process after SMService has been configured to no longer manage it. If only command line options have changed for a running process, you will need to stop the running process (with the old command line options still active) and call sms-start (with no options) to force SMService to restart the stopped process with the newly configured command line options. This is done so that SMService does NOT have to restart to reconfigure core NMS Services. If/when SMService is restarted WebLogic EJBs must fully re-cache and this will impact active NMS end users. By allowing NMS Service/Adapter configuration changes without having to restart, SMService can reduce the impact to active NMS end users.

The smsReport tool can be used to request and monitor the SMService view of the processes it is currently managing. smsReport is a non-GUI tool used to report the state of the system by querying SMService. It is executed in either one-shot or monitor mode. One-shot mode is the default mode that queries SMService for the current state and displays it to the user on exit. However, if the system state is INITIALIZING, then smsReport automatically switches to monitor mode so as to not exit prior to initialization completing

before exiting. Monitor mode is set by starting smsReport with the -monitor command line option. It serves the same function as the default one-shot mode but does not close after the system state has been reported.

To shutdown the Oracle Utilities Network Management System (gracefully) use the following script:

```
sms-stop
```

The `sms-stop` script will shutdown all of the user environments (one at a time) and then the services in reverse order to how they where defined to startup in the `~/etc/system.dat` file. Using this script generally prevents certain deadlock conditions which can occur if an attempt is made to stop all user environments and system services at the same time.

SMService can be configured to monitor any service or process, even non-isis aware processes as it does for lighttpd and the tao-naming service. To configure this, you will need to define a script that supports three parameter options: status, start, and stop. In addition to the parameters, the status should return zero if the process is not running, and 1 if the process is running. See `nms-lighttpd` and `nms-naming-service` product scripts as examples. The comments in the template system.dat file will describe how to configure the non-isis processes.

DBService - Database Service

DBService provides database access for any processes attached directly to the Isis message bus within the Oracle Utilities Network Manage System environment. The messaging backbone, Isis, directs database queries and commands to the appropriate Oracle RDBMS server and returns results to the requesting process.

> **Note**: A given instance of DBService allows a configurable number of queries to occur in parallel but serializes RDBMS updates. By assigning update responsibility of specific tables to specific DBService instances (by convention) parallel updates can be supported which generally increases performance and/or scalability under system load. TCDBService, MBDBService are examples of this strategy. Most major NMS Services (other than ODService) support the ability to leverage a dedicated DBService process. See the provided `system.dat.template` file for examples of this configuration.

ODService - Object Directory Service

ODService registers new objects as well as caches configuration and (optionally) run-time information that is likely to be requested by applications in a particular form and/or on a regular basis. This caching allows the requests to be handled very quickly without directly accessing the database. Cached information is primarily static configuration data, such as object classes, class hierarchy, symbology assignments and (optionally) device alias information.

DDService - Dynamic Data Service

DDService manages real time (dynamic) information required by the system. In addition to command line options DDService utilizes the srs_rules table for run time options.

Examples of dynamic data that DDService manages include:

- Current status of switchable devices
- Special operating conditions of devices (tags, crews, notes, etc.)
- SCADA information (analogs, digitals, quality codes)
- Operating authority (users and control zones)

DDService uses inheritance to determine which condition classes prevent device operations, and when. Condition classes that inherit from the "tag" base class always prevent device operations. Conditions that inherit from "block_if_hot" prevent operations only in energized areas, and conditions that inherit from "block_if_0" prevent operations when the condition is in status 0. "block_if_1" prevents operations when the condition is in status 1, etc. These condition statuses can be configured to change when safety documents transition to different states, such as Issued, Released, etc.

When you make changes to Oracle Utilities Network Management System control zones (control_zones and/or control_zone_structures tables), you need to tell DDService to update its internal control zone memory structures with the following UpdateDDS command:

```
UpdateDDS -recacheZones
```

When you make changes to SCADA device definitions, you can tell DDService to update itself with the following UpdateDDS command:

```
UpdateDDS -recacheMeasures
```

After a model build, recache zones and recalculate abnormal devices:

```
UpdateDDS -recacheAfterModelBuild
```

MTService - Managed Topology Service

Real-time electrical systems are in a constant state of flux of electrical flow. A single device operation could de-energize a model section, create a parallel on one or more phases, ground one or more phases, create a loop condition, or extend some other form of energization/deenergization. Since the topological state (energization, ground status, energizing feeder, feeder color, and so forth) of a device cannot be accurately determined without taking into account a large number of other devices and operating conditions, it is not possible for each application to independently determine current topological states. Instead, MTService maintains a complete topological copy of the model in memory, which it updates as devices and conditions change. It publishes topological impact updates and services topological data requests from other Network Management System applications and services.

JMService - Job Management Service

JMService is the customer trouble call analysis engine. It relies on MTService to trace device connectivity when determining probable outages in the system. Customer complaints (trouble calls) are fed into the system and JMService groups them using configurable rules to compute and publish the most likely cause of the problem. JMService also manages restoration resources (crews). In addition to command line arguments, JMService uses the srs_rules table for the majority of its run-time configuration options.

After a model build, recache control zone changes:

```
UpdateSRS -recacheZones
```

If disconnected customers have been added to the CES_DISCONNECTS table, refresh JMService:

```
UpdateSRS -recacheDisconnects
```

See Chapter 19 - JMService Configuration for more information.

TCDBService - Trouble Call Database Service

This is a copy of DBService that runs specifically to improve the performance of JMService by handling database calls for JMService. This lets the main DBService manage database requests from operator activity not directly related to trouble calls.

MBService - Model Build Service

MBService is used in building a data model, which mirrors the customer's existing data model (generally extracted from a Geographic Information System such as ESRI, Intergraph, SmallWorld, or AutoCAD). When changes are made in the GIS a project-specific extractor is used to extract and transform GIS changes into a standard Network Management System format. MBService takes the standardized input, parses and integrates the resulting changes into the Oracle Utilities Network Management System electrical network model. In addition to maintaining the model database, MBService also generates map files, which are optimized for use with Network Management System graphical viewing tools.

SwService - Switching Service

SwService manages the execution of VVO, FLISR, and Load Shed sheets. The execution of these sheets has to be managed by a service as a client is not required when VVO or FLISR are in Automatic mode or when the steps in Load Shed sheet are executed automatically. Even in Manual mode, SwService handles the execution of the switching sheet and takes over the single user lock on the sheet when the request is made. Users can load the sheet while it is being processed by SwService, but will only have View-Only access to the sheet. This service is also required to properly update switching sheet steps related to two way SCADA integration; particularly the completion of SCADA tagging steps and for reporting SCADA step execution failures. This service is only required to run when the **flisr**, **volt_var**, or **load_shed** modules are licensed or when two way SCADA integration is required for properly updating switching sheet steps.

MBDBService - Model Build Database Service

MBDBService serves the same purpose for MBService as TCDBService does for JMService. It is a copy of DBService that runs specifically to improve the performance of MBService by handling the database calls resulting from model building.

MQDBService - MQService Gateway DBService

MQDBService provides direct access to the database for the MQSeries Gateway. This reduces competing throughput for the DBService reserved for operator interactions.

PFService - Power Flow Service

PFService manages real-time operations power flow calculations that allow you to view the complex voltages and currents at points and devices in the electrical network model. These calculations are performed on an electrical island basis by tracing from each energized source and collecting all the energized objects. SCADA measurements at the feeder head and at various points down the feeder are used to accurately distribute load to each load point. PFService sends the real-time power flow solution results, as well as information about voltage and flow violations, to various Oracle Utilities Network Management System windows for you to view.

Note: PFService allows configurable numbers of processing tasks to occur in parallel, and, therefore, improve performance and avoid operator wait times. Two configuration rules drive parallel processing:

- **MAX_BACKGROUND_THREADS:** Sets the number of background tasks (FLM, FLISR, FLA, Real-Time Optimization) that can run in parallel. This number should be set to the number of threads/cores that are available for PFService background processing.
- **MAX_USER_THREADS:** Sets the number of user tasks (Suggested Switching, Power Flow Look Ahead, Study Mode Power Flow Analysis, and Study Mode Optimization) that can occur in parallel. It is recommended that this be set to one thread per five active DMS users - with a minimum of 10 allocated.

CORBA Gateway Service

The CORBA Gateway service provides part of the interface between the Java-based NMS user applications (Web Trouble, Storm Management, Web Call Entry,) and the other C++-based Oracle Utilities Network Management System services. The CORBA gateway allows the Java Application Servers to get published updates from services like JMService, DDService or MTService and also provides the mechanism to query these services directly on-demand. The WebLogic Java Application Servers must then take these updates and make them available for the Java (end-user) client applications.

The CORBA Gateway service uses Isis to communicate with the other Oracle Utilities Network Management System services. The CORBA Gateway service requires that the TAO (The ACE ORB) CORBA Naming Service be running. TAO is configured to run on startup via the sms-start script.

Note: We now recommend that you run three copies of the CORBA gateway for each Oracle Utilities Network Management System environment.

1. The first instance is a dedicated publisher instance that takes messages published via the Oracle Utilities Network Management System services and publishes them to the Java Application Server (WebLogic).
2. The second instance is dedicated to handling Java client application requests to Oracle Utilities Network Management System services. By default, client application requests to issue SCADA controls are rejected. To allow such requests, the instance handling client application requests must be running with the '`-scadacontrols`' command-line option.

Examples of how to setup these corbagateway instances can be found in the `$NMS_BASE/templates/system.dat.template` file.

3. The third instance is a generic publisher for publishing to email, and page. This is primarily used by Service Alert.

The WebLogic Java Application Server must be configured to correctly connect to the appropriate corbagateway(s). See the Oracle Utilities Network Management System installation guide for instructions on configuring the Java Application server.

Service Alert Service

Service Alert processes updates from other services such as job/event update information, device operations, as well as receiving notifications from database triggers. These "updates" serve as the triggers for Service Alert to determine when the criteria for sending out a notification have been met. Once triggered, Service Alert gathers relevant data and sends out the desired notifications.

Include Additional Customer Information into Notifications

Service Alert Service can load additional information from the CES_CUSTOMERS database table and include it in "Outage affecting specific customer(s)" and "Outage affecting critical customer(s)" notification messages. The **-cust_column** command-line option is used to specify a list of columns from the CES_CUSTOMERS table that should be loaded.

Format

```
-cust_columns [column name 1][,column name 2]...[,column name N]
```

where

`[column name x]` is the name of the column in the CES_CUSTOMERS database table

Example

```
-cust_columns city_state,zip_code
```

To include additional customer information into notification messages, you must edit the XSL documents that transform Service Alert Service generated raw XML into a user-friendly form. For example, the XSL fragment below appends city, state, and zip code information to the customer's address.

```
<td>
  <xsl:value-of select="concat(CUSTOMERADDRESS, ' ',
  CUSTOMEREXTRA/CITY_STATE,' ',CUSTOMEREXTRA/ZIP_CODE)"/>
</td>
```

XML Encoding Configuration

By default Service Alert Service sets encoding for the generated XML document to UTF-8. The **-encoding** command line option allows you to specify a different encoding to be used. To avoid XSL transformation issues and ensure that the data is rendered correctly, the XML document encoding should be compatible with the character set of the data loaded from the database (check the NLS_LANG environment variable).

Format

```
-encoding [XML encoding name]
```

Example

```
-encoding ISO-8859-1
```

Include Device Attributes Information in Notifications

Service Alert Service can load device attributes from the DEVICE_ATTRIBUTES database table and include it in *Outage*, *Outage affecting specific customer(s)*, *Outage affecting critical customer(s)*, and *Frequency of outages on a device* notification messages.

The -devattr command-line option is used to load device attributes from the DEVICE_ATTRIBUTES table.

Format

```
-devattr
```

Wait Period After Device Operation

Intelligent switching devices in the electrical network (for example, Automatic Throw Over switches) can cause a single device operation to be followed by several other device operations executed automatically. Service Alert would normally generate notifications as soon as an outage matching at least one of the configured notification parameters is detected. The -devopwait command line option allows real device outage notification processing to be delayed by the specified number of seconds. Adding a delay permits the electrical network to reach a stable state so that generated notifications would have accurate information.

Format

```
-devopwait [wait period in seconds]
```

Example

```
-devopwait 30
```

Enable Crew-Related Notifications

Service Alert supports generating of notifications when outage-related crew activities occurs, such as when the crew is assigned, en-route, onsite or suspended. This support is disabled by default, but can be enabled by running Service Alert service with the **-crew** command line option.

Note: Enabling this option does have a negative effect on performance.

Format

```
-crew
```

Service Alert Email Administration

How Service Alert Email and Paging Notification Work

When initiating a notification, Service Alert sends email and paging requests to the genpublisher service. It is the email toolkit code within the genpublisher that interfaces with a mail system. The email toolkit uses SMTP to send these message requests. * Therefore, to properly receive Service Alert notifications, an SMTP server needs to be configured and running on the network. All that is left to do is to describe to the email toolkit the configuration settings that it needs in order to communicate with the SMTP server.

> **Note:** Pager notifications are also sent by SMTP, since most major paging providers allow messages to be sent to a pager via an email aliasing system.

Entering Email/Pager Configuration Settings

The following Unix environment variables need to be set up properly in order to configure the email/pager notifications.

Variable	Description
NMS_SMTP_SERVER	This is the fully qualified network hostname of the mail server.
NMS_DOMAIN_SUFFIX	Domain Suffix – This value should be a valid domain such as "oracle.com". This value is used in constructing the domain portion of the "From" field for all outbound messages. This field is also used during SMTP communication between the CORBA gateway and the mail server. It is important to set this to a valid domain, as some SMTP servers will verify that the domain exists and is real. If the server does not believe that the domain is legitimate, the email message may be discarded

The **Email Username** setting is a command line parameter on the genpublisher service. The username is the string that appears after the "-username" command line option. This will appear in all email and pager notifications **From** field. It is probably a good idea to set up an email alias for this username, in case notification recipients attempt to reply to a notification. Note that the "@domain.com" portion of the username should be omitted as this comes from the "NMS_DOMAIN_SUFFIX" environment variable.

Verify in the genpublisher log that the change is in place by locating a line similar to:

```
01/23 12:59:59: CorbaGateway ctor: changed USERNAME from "nmsadmin" to
[some name]
```

Email Subject Line Configuration

The email and text/SMS notification messages contain a subject line. The subject line content is configurable via the MYC_MSG_SUBJECTS database table.

Definition of the MYC_MSG_SUBJECTS Table

Column	Description
MSG_SUBJECT_ID	Primary key. Assigned from the MYC_MSG_SUBJECTS_SEQ database sequence.
MESSAGE_TYPE_ID	Notification message type. See database table MYC_MESSAGE_TYPES for the list of valid notification message types.
TRIGGER_CODE	Notification trigger code. Valid values: 1 - Outage started/in-progress 2 - Outage restored 3 - Planned outage is scheduled 4 - Outage is canceled 5- ERT has been updated for an outage 6 - Storm started 7 - Storm ended 8 - Storm has been updated 9 - Crew action has occurred for an outage Note: The list of applicable trigger codes differs depending on the notification message type.
PLANNED	Planned/unplanned outage notification. Valid values: Y - planned N - unplanned
EMAIL_SUBJECT	Subject line for email notification messages.
PAGER_SUBJECT	Subject line for text/SMS notification messages.

When populating EMAIL_SUBJECT and PAGER_SUBJECT columns substitution variables can be used to include certain values from the notification message into the subject line. The following table lists available substitution variables.

Name	Description
CTRLZONE1	Name of the level 1 control zone
CTRLZONE2	Name of the level 2 control zone
CTRLZONE3	Name of the level 3 control zone
CTRLZONE4	Name of the level 4 control zone

Name	Description
CTRLZONE5	Name of the level 5 control zone
CTRLZONE6	Name of the level 6 control zone
EVENTIDX	Outage event index
EXTERNALID	Outage external id
DEVICEALIAS	Outage device alias
DEVICECLASSNAME	Outage device class name
CUSTOMERSOUT	Number of customers affected by the outage (system-calculated value)
USERCUSTOUT	Number of customers affected by the outage (user-provided value)
CRITICALCUSTOMERSOUT	Number of critical customers affected by the outage
WCUSTOUT	Weighted count of customers affected by the outage
SHEETNUM	Switching sheet number
RELATEDEVENT	Event index of the related event
RELATEDTYPE	Relation type code. Possible values: N - nested M - momentary P - partial restoration L - lead event of partial restoration M - manual S - switching

In addition, generic event fields can be used as substitution variables. In this case, the generic event field name serves as the substitution variable name. To use a substitution variable when configuring the email or text/SMS subject line, its name needs to be prefixed with dollar sign (for example, `$CTRLZONE1`).

Service Alert Script Configuration

Service Alert has the ability to deliver notifications via scripts. The notification content is passed to the script as standard input. The script is expected to return 0 in case of a successful execution. The script is executed by the Corba Publisher process, which is used to deliver Service Alert notifications.

The list of scripts available to Service Alert is stored in the PUBLISHER_SCRIPTS database table.

Column	Description
SCRIPT_ID	Unique script identifier. This value will be displayed in the Service Alert Administration tool.
APP	Application that uses this script. For Service Alert, this should be `service_alert`.
PATH	Full path to the executable.
DESCRIPTION	Script description.

Example of Adding a Script

```
INSERT INTO publisher_scripts (script_id, app, path, description)
VALUES ('cat', 'service_alert', '/bin/cat',
        'Print input to standard output');
```

Services Configuration File

The Services Configuration Data File (system.dat) configures services for operation. It determines how services are defined, which default flags to use, on which computers, and how long the waitfor timer runs. The system.dat file is located in the $NMS_HOME/etc directory.

There are a number of sections in the system.dat file. The most critical sections include:

- scripts
- server
- services
- applications
- program
- instances

Scripts

The following table defines the scripts that SMService uses to perform various tasks.

Script	Description

LaunchScript	Used to launch a service. The most widely used mechanism for starting all the services is:
	`sms-start`
	
	The default script to start a single service is `sms-start-service`. Its syntax is:
	
	`sms-start-service [host] [service] [process] [options]`
	
	• **host:** Name of the machine on which to run the service
	• **service:** Name of the service
	• **process:** Name of the executable that launches the service
	• **options:** Command line options that are passed to the process at initialization
	
	For example, to start DBService, type:
	`sms-start-service train1 DBService DBService -nodaemon`
	
	Define the launch script in system.dat as follows:
	`LaunchScript [script name]`
	
	If no script is specified, then `sms-start-service` is assumed.

Script	Description

Notify Script	Announces an event. This script eliminates the need for an Isis tool as an announcer. It can be used to generate e-mails and logs, or to interface to paging systems. When developing this script, keep in mind that it does not connect to Isis. The syntax is:
	`[script name] [time] [host] [process] [event type] [system state] [old system state] [message]`
	• **time:** Date/time stamp.
	• **host:** Name of the machine on which the processes are running.
	• **process:** Name of the process.
	• **event type:** The process state. Valid values are:
	  • STARTING: The process has started.
	  • INITIALIZING**:** The process has registered and is initializing.
	  • RUNNING: The process reports as initialized.
	  • FAILED: The process has failed.
	  • FAILED_INTERFACE: The process reports a failed interface.
	  • STOPPED: The process intentionally stops.
	  • INFO: The process generates a progress report.
	• **system state:** State of the system. Valid values are:
	INITIALIZING: SMService is launching processes from system.dat.
	NORMAL: All processes are running or are intentionally stopped.
	WARNING: A non-critical process has failed. This state also refers to failed critical processes that have another instance running.
	CRITICAL: A critical process has failed and there are no other instances running.
	• **old system state:** State of the system before the event generating the announcement occurred.
	• **message:** Message supplied by SMService or the process that caused the event.
	Define the notify script in system.dat as follows:
	`NotifyScript [script name]`
	There is no default value, so if a script is not defined here, then only Isis announcements are generated.

CoreScript	SMService looks to this script for instructions when a core file is detected. This script determines what should be done with the file, such as announce the existence of the file, delete it, archive it, or e-mail the administrator. It does not connect to Isis. Its syntax is:
	`[script name] [process] [corefile]`
	process: Name of the process that has produced the core file
	corefile: Path to the core file
	Define the core script in system.dat as follows:
	`CoreScript [script name]`
	If a script is not defined, the core file remains and will be detected by SMService during the next cycle.

Example Scripts

The following scripts demonstrate defaults/examples and show how SMService can be extended.

Name	Description
sms-notify	This is a NotifyScript and can be configured as the notify script in system.dat (this example is very simple), it merely calls the notify by email script.
sms-notify-by-email	This is a somewhat more advanced script for sending notifications of selected events to a group of administrators by email. The script uses the information supplied by SMService on the command line to determine if this event is putting the system into a critical state - or is clearing a previously existing critical state. If either of these two events are detected an email is constructed and sent to the appropriate people. The script is quite configurable and can be easily modified to change the names of the recipients, the mailer program used to generate the email, etc. This script can be configured directly as the NotifyScript in system.dat, or it can be invoked by an intermediate script such as `sms-notify`.
sms-core-save	This is an example of a CoreScript. It simply moves the core file to an archive directory and renames it based on machine name, process name and timestamp. It then generates an email to the administrators to inform them that a core file was found.
sms-start	This script is the way you start the NMS services. SMService is launched and then smsReport is run in one-shot mode. This means that the script will run until SMService starts reporting a system state as something other than initialized sms-start uses the Action command to determine if SMService is already running on this machine. If it is, it simply sends it a restart message.

Server

This section of the system.dat file defines all machines that run services. Each server must be assigned a separate server ID number from 1 to 10. The format is:

```
service [hostname] [server id]
```

For example, for services running between machines london and paris:

```
server london 1

server paris 2
```

The value for hostname can be specified literally as <local>. If this is the case, then SMService will automatically substitute the name of the current node as the machine name. For example:

```
server [local] 1
```

Support for running NMS Services on multiple nodes is a legacy option; the only practical and supported option is to use the "server [local] 1" type configuration.

Service

These entries in the system.dat file are definitions of services and process groups, such as interfaces, that are launched and monitored by SMService. Below is a sample service section:

# NAME	REQUIRED	START	DELAY	RESTARTS	RESET	MODE
service SMService	Y	60	0	10	86400	
service DBService	Y	90	0	10	86400	
service ODService	Y	180	0	10	86400	
service DDService	Y	180	0	10	86400	
service MTService	Y	180	0	10	86400	
service MBService	Y	180	0	10	86400	
service JMService	Y	280	0	10	86400	
service SwService	Y	280	0	10	86400	
service PFService	Y	4000	0	10	86400	
service corbagateway	Y	120	0	10	86400	
service service_alert	Y	120	0	10	86400	

The following table describes the SMService Service fields.

Field	Description
NAME	The name of the executable for the particular service.
REQUIRED	Indicates whether the instance of the service is required for the system to be functional. Valid values are 'Y', 'Yes', 'N', or 'No'. If there are no instances of a required service, the system locks until an instance is started.
START	The time taken for a service to start.
DELAY	Sets the number of seconds to wait before restarting a failed service. It only applies to processes that failed after they were running. Processes that fail before initialization are restarted based on the period parameter. A negative number indicates that the process is not restarted.
RESTARTS	The number of times to attempt restarting a process. A process is no longer automatically restarted after this value is exhausted until the process is reset (see below).
RESET	The timeout period that controls the rate at which processes are reset. When a process is reset, the restart counters re-initialize. A negative value deactivates this feature.
MODE	An optional argument that specifies the high availability mode of the service. If a mode is specified, the service starts with `-[mode]` and `-number [n]`, where `[n]` is the id defined for the node in the server line. Valid modes are exclusive, redundant, parallel or not specified. Exclusive runs with only one server. Redundant specifies running two servers, each with a database that mirrors the other. Parallel involves using Oracle Parallel Server to run two servers with a shared database.

Program

The program section of the system.dat file defines the executable program and command line options for each service. This section is optional, but can be used for the following:

- Specifying an alternative executable for a particular service. For example, setting TCDBService as an instance of DBService.
- Specifying command line options across all instances of a service. This simplifies the instance definition so that the command line options do not have to be duplicated for each definition.

Below is a sample applications section:

# NAME	EXE	ARGS
program DBService	DBService	-nodaemon
program ODService	ODService	-nodaemon -aggregates
program DDService	DDService	-nodaemon -zones -subscribezone -allowReset -alarms ALL
program MTService	MTService	-nodaemon
program MBService	MBService	-nodaemon
program JMService	JMService	-nodaemon -dbs
program SwService	SwService	-nodaemon
program PFService	PFService	-nodaemon
program corbagateway	Corbagateway	-nodaemon -ORBInitRef NameService=iioploc://[hostname]:1750/NameService -ORBLogFile /users/[username]/dialog_log/orb.log -ORBDebugLevel 3 -implname InterSys_[hostname]_[username] -iorfile /users/[username]/etc/[username]_vns.ior -publisher -xmldir /users/[username]/dist/wwwroot/xml
program service_alert	Mycentricity	-nodaemon -xmldir /users/[username]/dist/wwwroot/xml

The following table describes the SMService Program fields.

Field	Description
NAME	Specifies the name of the service that the executable belongs to. Valid services for this value are defined in the service section.
EXE	Specifies the name of the executable that runs the service.
ARGS	Defines the command line options that are used in all instances of the service.

displayName

The displayName section of the system.dat file defines the display name for executable programs. This section is optional, but can be used to define user-readable names for the executables.

Below is a sample displayName section:

```
#           NAME          DISPLAY NAME
displayName SMService     System Monitor Service
displayName DBService     Database Service
displayName TCDBService   Trouble Call Database Service
displayName MBDBService   Model Build Database Service
displayName ODService     Object Directory Service
displayName DDService     Dynamic Data Service
displayName MTService     Managed Topology Service
displayName MBService     Model Build Service
displayName JMService     Job Management Service
```

Instance

The instance section of the system.dat file defines how the services are started. The format of each line is:

```
instance [node] [service] [database/args]
```

The following example starts nine services on the local node.

#	NODE	SERVICE	DATABASE/ARGS
instance	[local]	SMService	
instance	[local]	DBService	
instance	[local]	ODService	
instance	[local]	DDService	
instance	[local]	MTService	
instance	[local]	JMService	
instance	[local]	SwService	
instance	[local]	corbagateway	
instance	[local]	service_alert	

The following table describes the SMService Instance fields.

Field	Description
NODE	Defines the node. Valid nodes for this value are defined in the server section. The value for NODE can be specified literally as `[local]`. If this is the case, then SMService will automatically substitute the name of the current node as the instance for which the service is to be started. By using `[local]` in place of a specific machine name, you can simplify your effort when replicating a system; you will not need to make changes to the system.dat at all.

SERVICE	The service being defined.
DATABASE/ARGS	Command line arguments that are applied when the service starts at this node. If the program section specifies command line options for a particular service, it applies to all nodes, so the arguments do not need specification here.

See Appendix A - Command Line Options and High Level Messages for service command-line options and Action commands.

Model Build System Data File

The Model Build System Data File (system.dat.model_build) configures services for Model Build/Configuration operations. It is formatted the same as system.dat.

The system.dat.model_build starts only SMService, DBService, ODService, and MBService. These services are generally executed from configuration scripts, such as nms-setup, which require that some services be running to access the database and object classes.

Starting and Stopping Services

In order to start services, the following configuration files must be updated for the specific site configuration:

```
~/etc/system.dat.model_build

~/etc/system.dat

~/etc/system.dat.init
```

Starting Services

To start services, complete these steps:

1. Login to the server machine as the Oracle Utilities Network Management System Admin user.
2. Enter:

   ```
   sms-start
   ```

   SMService starts. It reads and caches the system.dat file by default and starts the remaining services based on the data it just cached.

   Note: Using the `-f [filename]` option with sms-start will override the default behavior and SMService will cache the specified file instead (for example, `~/etc/system.dat.init` or `~/etc/system.dat.model_build.etc`).

Stopping Services

- To stop clients:

  ```
  sms-stop -c
  ```

  Stop all user sessions with the -c option, which notifies Web Workspace clients that the administrator is forcing an exit. The session stops immediately if users acknowledge the dialog box or automatically after a 5 second delay.

- To stop services:

  ```
  sms-stop -s
  ```

  When stopping services, you may have other tools running. The services are the core dependencies of all applications, so when services are stopped, all tools should be stopped and then restarted after the services have been re-launched.

- To stop both clients and services:

  ```
  sms-stop -a
  ```

  Note: Occasionally, there are tools or Isis processes that may continue to exist as defunct and/or hung processes after the above commands do (or do not) run to completion. Check the process list on the Unix machines for these processes and kill them prior to restarting. Otherwise, otherwise the system may not restart properly.

- To set all users as logged out in the CES_USER_LOG table:

  ```
  sms-stop -u
  ```

  Note: this should only be done when all users are told to stop their processes.

Starting and Stopping the WebLogic Managed Server

Additional scripts are provided to start and stop the WebLogic managed server individually and in parallel with starting or stopping services. These require configuration from the nms-wls-config script.

nms-wls-config

Generates the configuration files required for nms-wls-control, nms-all-start, and nms-all-stop. This scripts prompts for WebLogic Admin Server credentials. If not provided on the command line, it prompts for the admin server URL and managed server name as well. Credentials are stored encrypted in $NMS_HOME/etc/wls in the format used by the WLST storeUserConfig function.

Usage: `nms-wls-config [-a ADMINSERVER] [-s SERVER]`

Configure WebLogic Connect

Optional arguments:

```
-h, --help            show this help message and exit
-a ADMINSERVER, --admin ADMINSERVER
                      URL of the adminserver. This is of the form
                      t3:/hostname:7001
                      t3s:/hostname:7002
-s SERVER, --server SERVER Name of the managed server
```

nms-wls-control

Uses the WebLogic Scripting tool to start or stop the WebLogic managed server associated with an NMS environment. This requires configuration created by the nms-wls-config script.

Usage: `nms-wls-control start | stop`

nms-all-start

Starts NMS services and the WebLogic managed server in parallel using sms-start and nms-wls-control. This requires configuration created by the nms-wls-config script.

Usage: `nms-all-start`

nms-all-stop

Stops NMS services and the WebLogic managed server in parallel using sms-stop and nms-wls-control. This requires configuration created by the nms-wls-config script.

Usage: `nms-all-stop`

Chapter 8

Building the System Data Model

The Model Build process creates the operations data model that mirrors the utility company's Geographic Information System.

This chapter defines the configuration of the model builder and provides an overview of validating and testing tools. It includes the following topics:

- Model Builder Overview
- Data Directories
- Model Configuration
- Customer Model - Logical Data Model
- Customer Model Views
- Model Build Process
- Model Manipulation Applications and Scripts
- Schematics
- In Construction Pending / Device Decommissioning (ICP)
- Auto Throw-Over Switch Configuration (ATO)
- Symbology
- DMS Data Model Requirements and Configuration
- Catalog Tables
- Power Flow Service High Level Messages and Command Line Options
- Spatially Enabling the Data Model for Outage Analytics
- NMS CIM Import and Export Tools
- Model Build File Export to XML

Model Builder Overview

The Model Builder Service (MBService) is used in building an Oracle Utilities Network Management System operations data model. The Oracle Utilities Network Management System operations data model is built using the customer's existing *as-built* data model, which is typically a Geographic Information System (GIS) or graphic files (*e.g.*, AutoCAD). There are options to bring in construction information to support commissioning and decommissioning of model data, referred to as In Construction Pending (ICP). Necessary enhancements are applied to the GIS data model to make the *real-time* data model.

When changes are made in the GIS, MBService then merges them into the Oracle Utilities Network Management System data model. In addition to maintaining the model database, MBService also generates map files that are loaded for visual inspection.

A single spatial grouping of data known as a partition passes through various stages during its incorporation into the Oracle Utilities Network Management System Operations Model:

- GIS Data Extraction – to extract the data from the GIS to Oracle's vendor neutral model preprocessor (*.mp) file format.
- Preprocessing – to produce model build (*.mb) files used by the Model Builder.
- Model Build (MBService) – saves the information into the Oracle Utilities Network Management System Operations Model RDBMS and writes out a set of maps.

The Model Builder service (MBService) is responsible for managing structural changes to the core operations model. Structural changes are largely the creation, deletion, and modification of objects. Non-structural changes involve updating attribute information such as status values.

The core operations model describes a set of interconnected network components with graphical representations and managed statuses. The objects contained within the model are subdivided into partitions with interconnections of partitions managed through the use of boundary nodes.

This data model must initially be obtained from an external source (such as a GIS) to populate the core operations model. Once populated, the core operations model is the basis for support of system services and the construction of diagrams.

The real-time services typically load parts of the model during initialization. These services also update attributes of the model. The process of model edit involves the creation, update, and deletion of objects that require consequential updates within services.

Patches

Import (*.mb) files are submitted to MBService for processing. Each set of import files submitted to MBService is considered a model patch and is applied to the current model. Most often, a patch is generated when a single partition (often data that represents an electrical circuit/feeder or similar) is submitted to MBService for building.

The lifetime of a patch includes the following:

- Initial creation of the patch either locally or externally.
- Addition of the patch to the core operations model, where the patch will either be applied and become part of the current operations model or will be deleted if there is a problem with the patch resulting from patch format errors or real-time issues in the operations model (such as deleting a device with an active call or outage).

Data Directories

OPERATIONS_MODELS Directory

The `$OPERATIONS_MODELS` environment variable points to a file system directory (often `${HOME}/data`). This directory must be writable by the Oracle Utilities Network Management System Model Build Service (MBService). One significant use of this directory is as a home for the graphical meta-files (`*.mad` and `*.mac`) that MBService writes for each model import (`*.mb`) it processes. It contains the model map files that the Web Application Server reads and makes available to the Viewer, which presents them to the operator. It also contains the files associated with the model build process such as preprocessor import files, model build import files, and log files.

The `$OPERATIONS_MODELS` directory typically has the following structure:

OPERATIONS_MODELS/
- **SYMBOLS/**
- **drawings/**
- **errors/**
  - `Patch[n].log`
  - `[map].log`
- **mp/**
  - `*.mp *.mpd`
  - **done/**
    - `*.mp *.mpd`
- **patches/**
  - `*.mb *.mbd`
  - **done/**
    - `*.mb *.mbd`
- **reports/**
- `*.mad`
- `*.mac`

The following table describes the Model Builder directories and files.

Directory/ Files	Description
SYMBOLS	Contains the defined symbol sets for the presentation of all objects. (Convention only. May be moved elsewhere.)
drawings	Contains the drawing and documentation files that can be associated with objects in the model. (Convention only. May be moved elsewhere.)
errors	Contains the output files of the model builder specifically related to errors and patch processing. The Model Build patch log files are named in `Patch[patch_number].log` format. Preprocessor map log files are typically named in `[map_name].log` format.
*.mac	Textual representations of the background maps. The background map files corresponding to the *.mad files. These files are used by the Viewer to present background graphic information (boundaries, roads, text, etc.).
*.mad	Textual representations of the electrical maps. The map files used by the Viewer when presenting graphic information correlated to the network information stored in the database. It is essential to keep the database and the maps synchronized to ensure proper presentation and map conductivity.

Directory/ Files	Description
mp	The mp directory is the location of model preprocessor import files, `[mapname].mp` or directories, `[mapset].mpd`. This is project dependent however most projects will import `.mp` files.
patches	Contains `[mapname].mb` files and/or `[mapname].mbd` directories. These files define the model build transactions that will be submitted to the model. Files are moved into the done subdirectory after they have been submitted.
reports	This directory contains difference reports, which list all changes being introduced into the model for each patch. These are only generated if MBService is running with the `-report` option.

Model Configuration

Model configuration requires a number of configuration parameters, scripts, and SQL files to be defined in order to fully set up an operational Oracle Utilities Network Management System. The following sections provide a checklist of the configuration settings that are required for a successful model build.

Define Environment Variables

Each user of the Oracle Utilities Network Management System must have an **.nmsrc** file in the $NMS_HOME directory. Edit the **.nmsrc** file toset the following required environment variables:

Environment Variable	Description
DATEMSK	This environment variable points to the path of the nms_datefmt file.
NMS_ROOT	This environment variable is set to the directory where the top of the Oracle Utilities Network Management System installation occurs (`users/nmsadmin`).
NMS_HOME	This environment variable is set to the home directory of the nmsadmin user; it is the same as $NMS_ROOT (`users/nmsadmin`).
NMS_BASE	This environment variable is set to the product installation directory (for example, $NMS_HOME/ product/2.5.0.0).
NMS_DATA_FILES	This environment variable is set to the directory where most configuration data files used by Oracle Utilities Network Management System software are installed. This includes *.dat, *.sym, *.cel files, among others.
NMS_DATA_TABLESPACE	Contains the name of the primary Oracle tablespace. The installation and setup process uses it to better manage how database tables are set up.

Environment Variable	Description
NMS_INDEX_TABLESPACE	Contains the name of the Oracle tablespace that is to be used for most indexes. The installation and setup process will attempt to put most indexes into this tablespace. It is not currently practical for an Oracle RDBMS Data Definition Language (DDL) statement for a create table operation to place the primary key (for the table in question) into an alternate tablespace. Most modern Oracle RDBMS installations use either solid state storage or some kind of complex Storage Area Network (SAN) for physical storage. These types of systems do not tend to significantly benefit (from a performance perspective) separating Oracle table and index data into separate tablespaces. As a result this option can be and is often set to $NMS_DATA_TABLESPACE
NMS_LOG_DIR	This environment variable is set to the location of the service log files.
NMS_CONFIG_ORDER	This environment variable contains a list of a set of configuration standards. Oracle defines the standard base configuration upon which customer configurations are built. The NMS_CONFIG_ORDER variable indicates which configurations to use. The setup process looks only for the files containing the values specified in this variable. The syntax is: `NMS_CONFIG_ORDER = "[project] nms"` The first argument is the name of the customer or project. The last argument is the name of the default base configuration. There may be multiple configurations specified between the first and last arguments. When the system boots it processes the arguments from right to left, so it first loads the base configuration. Then it moves on to the previous argument and loads the associated configuration files if they exist. The process continues until each argument is processed.
NMS_SMTP_SERVER	This environment variable points to an SMTP server where mail transactions can occur.
NMS_SQL_FILES	This environment variable is set to the directory where most SQL files used by Oracle Utilities Network Management System software are installed.
OPERATIONS_MODELS	This variable specifies the directory into which the model will be built. That is, all maps and log files from the model build are located in this directory.
RDBMS_ADMIN	The tnsname for establishing a sqlnet connection via Oracle to the admin schema. This value must exist in the tnsnames.ora file on the system attempting a connection. This is required for the use of many setup scripts and `ISQL -admin`. This value must be different from RDBMS_HOST.

Environment Variable	Description
RDBMS_HOST	The tnsname for establishing a sqlnet connection via Oracle to the read/write schema. This value must exist in the tnsnames.ora file on the system attempting a connection. This is required for the use of many setup scripts and ISQL. This value must be different from RDBMS_ADMIN.
RDBMS_TYPE	Database type (ORACLE_OCI).
SYMBOLOGY_SET	This environment variable is set to the full path of the Oracle Utilities Network Management System symbol file `[project]_SYMBOLS.sym`. **Note:** This is mostly a legacy method for graphical symbol definition. More recent configurations tend to rely more on Scalable Vector Graphic (SVG) symbols (1 SVG per file).

Configure Isis

Isis is the messaging backbone used by the Oracle Utilities Network Management System Operations Model, and it is required for every step of a model build. See the Isis Configuration chapter for information about setup and configuration.

To ensure Isis is running, type:

```
nms-isis status
```

Result: If Isis is running, a "1" is returned, but (as is accepted *nix practice) the return value of nms-isis process itself is 0 (echo $?=0). If Isis is not running, a "0" is returned but $?=1. This can be useful in common shell scripts used to monitor/manage the Isis message bus.

Verify Database Connection

Through the installation process the nmsrc file and the Oracle Wallet should be setup correctly so the ISQL script can be run by an administrative user to connect to an interactive session of the database.

ISQL can make a connection to the database. To verify that a connection is possible to the database, complete these steps:

1. From the [project] user name on the master server, enter:

   ```
   ISQL
   ```

   A database prompt ensures that the environment is set up correctly.

   This schema/user should not have Data Definition Language access - only Data Manipulation Language access.

2. Enter quit to exit the database connection.

3. From the same [project] user name on the master server attempt to access the admin schema/user for Data Definition Language access:

   ```
   ISQL -admin
   ```

   A database prompt ensures access to the Oracle RDBMS admin schema for Oracle Utilities Network Management System.

4. Enter quit to exit the database connection.

Directory Set Up

The model builder is primarily concerned with the tables within the selected database and the directory structure located under ${OPERATIONS_MODELS} as shown below.

Setting up the Directory Structure

If the directory structure has not been set up, run the script nms-mb-setup to configure it. It requires the OPERATIONS_MODELS environment variable to be set to the user's map data directory.

> **Note:** The nms-mb-setup script is part of the model setup process, so the step listed here is redundant if this has already been completed.

The [project]-mb-setup script creates and cleans the directory structure for customer specific model build setups.

The [project]-mb-preprocessor script is called during the initial setup process to set up any additional directories or database tables that may be required by the model preprocessor. It is only required if special setup is needed.

Cleaning Up the Directory

If the data directory already exists from an obsolete data model, nms-mb-setup -clean should be called to clean up all the residual files.

> **Note:** If you run this script with the -clean option, you will **delete** the operational model.

Define and Organize Classes

The operations model is designed around a class hierarchy. At the top of the hierarchy is the superclass, from which all other classes inherit attributes. The hierarchy may have multiple levels, each level having a parent/child relationship. The superclass is the only level that is always a parent and never a child.

Class Inheritance Definition

Classes and inheritance are defined and configured in the [project]_classes.dat and [project]_inheritance.dat files, respectively, located in the [project]/data directory. These files are loaded when the nms-setup command is run to set up the data model.

These files can be individually loaded using the ODLoad command. The syntax to load classes.dat in Classes table via ODLoad is:

```
ODLoad -c [filename]
```

The inheritance relationships file, inheritance.dat, can be loaded into the INHERITANCE table via ODLoad. The syntax is:

```
ODLoad -i [filename]
```

In addition to these base class and inheritance files, special files may be included for dynamic condition classes ([project]_cond_classes.dat, [project]_cond_inheritance.dat) and classes required for the power flow application ([project]_pf_classes.dat, [project]_pf_inheritance.dat). These additional files would be supplemental to the base files and should not duplicate any entries.

Oracle includes some required classes within the nms_core_classes.dat file. These classes are required in order for the Oracle Utilities Network Management System to work properly. Their inheritance is defined in nms_core_inheritance.dat and is also required. None of the information in these files should be changed, removed, or duplicated.

Configure Attribute Table

The Oracle Utilities Network Management System attribute table is populated using [project]_attributes.sql. The user attribute table is populated using the [project]_model_attributes.sql file.

Configure Control Zones

Control Zones are discrete, hierarchical sections of a utility's distribution system. Control Zone configuration requires defining zones, assigning devices to zones and, optionally, creating zone sets (or groups) that assist in assigning crews to multiple zones and to filter crews.

Configuring Electrical Devices

If you plan to use the Oracle Utilities Network Management System control authority functionality, then all electrical devices should have an assigned Network Component Group (NCG). This is usually assigned in the source data or computed in the preprocessor.

Defining Control Zones

Control Zones are initially defined in the [project]_control_zones.dat file, which is a text file that is read by nms-control-zones as part of the nms-setup and, generally, by the [project]-postbuild script following model builds. Product configuration uses a depth of five control zones, but you can configure zones to a maximum depth of 10 levels.

Defining Control Zone Sets

Control Zone Sets may be defined for crew assignments. Control Zone Sets are defined in the control_zone_set table.

To create Control Zone Sets, do the following:

1. In Oracle SQL Developer (or an equivalent application), search the **control_zones** table for the NCG codes of the zones that you will be adding to sets.
2. Add one row in the **control_zone_set** table for each control zone in the set.
3. Reload DDService:

   ```
   UpdateDDS -recacheZones
   ```

Example

For example, you want your zone set (**SuperZone Set**) to have two zones (**Northern Region** and **Fuzzy**). After checking the control_zones table, you determine that Northern Region has an ncg_id of 100000130 and Fuzzy has an ncg_id of 1.

Using a sql file to add the rows to the control_zone_set table, you add the following sql commands:

```
insert into control_zone_set ( set_name, ncg_id, description ) values
( 'SuperZone Set', 100000130, 'Northern Region Test' );
insert into control_zone_set ( set_name, ncg_id, description ) values
( 'SuperZone Set', 1, 'Northern Region Test' );
commit;
```

If you want to restrict the zone set visibility to certain users only, add records to the `CONTROL_ZONE_ALIASES` table for those users only. If no `CONTROL_ZONE_ALIASES` records are present for that zone set, the zone set will be available to all users. If you want to exclude access to zones in the zone set from certain users, add entries in the `EXCLUDED_ZONE_SET` for the set_name value used above. Similarly, add records to the `DEFAULT_ZONE_SET` table to give users automatic access to them.

Configure Symbology

Oracle Utilities Network Management System Viewer symbol information is stored in [project]_SYMBOLS.sym file for legacy soft symbols or in the $NMS_CONFIG/jconfig directory structure for Pixmap or Scalable Vector Graphics symbols. The [project]_ssm.sql file maps classes to the particular symbol. The symbology file build process has been standardized to build the run-time symbol file ($NMS_CONFIG/data/SYMBOLS/[project]_SYMBOLS.sym) from these symbol file sources in order of increasing preference:

1. `$NMS_BASE/product/data/SYMBOLS/MASTER_SYMBOLS.sym,`
2. `$NMS_BASE/i18n/data/SYMBOLS/MASTER_SYMBOLS.sym,`
3. `$NMS_CONFIG/data/SYMBOLS/[project]_DEVICE_SYMBOLS.sym,`
4. `$NMS_CONFIG/data/SYMBOLS/[project]_CONDITION_SYMBOLS.sym.`

The command, `nms-make-symbols`, will do the construction of the run-time symbology file and will make a backup of the resulting file if one existed prior to the execution of this script. Run nms-make-symbols before running nms-install-config to get your `[project]_SYMBOLS.sym` file up to date with the your latest configuration and NMS product release.

Service Configuration File

The `sms-start` script is used to start up Oracle Utilities Network Management System services. It normally reads the system.dat file to determine which services to start up and what arguments to give them. Before a model is built, this configuration must not be used, because it contains startup commands for the Dynamic Data Service (DDService), the Managed Topology Service (MTService), and the Job Management Service (JMService), none of which will execute until a model has been at least partially built. The model build process expects to find another configuration file, system.dat.model_build, in the same directory that has a more limited set of services. In addition, the `system.dat.init` file starts up the database services (DBService and ODService) that are used for initial setup.

Verify Licensed Products File

The Automated Setup script (`nms-setup`) and related *.sql and script files will reference a `[project]_licensed_products.dat` file to properly configure the model to support the products you have licensed. This file is a text file and contains a list of the licensed Oracle Utilities Network Management System options. There is a template version of this file in `$NMS_BASE/templates/licensed_products.dat.template`. The template should be copied to your `$NMS_CONFIG/sql` directory and renamed to a *`[project]`*`_licensed_products.dat` file. Then you should edit the file to uncomment the options you have licensed and are implementing. This edited template file should then be installed using the `nms-install-config` installation script prior to running the `nms-setup` command.

The following table describes the product codes used in the template file; refer to the template file for the most current set of product codes and descriptions.

Product	Description
amr	Generic MultiSpeak AMR/AMI integration
bi	Oracle Utilities Analytics integration
fla	Fault Location Analysis
flisr	Fault Location, Isolation, and Service Restoration

Product	Description
flm	Feeder Load Management
ivr_gateway	Generic Interactive Voice Recognition integration
load_shed	Load Shed and Restoration
mobile	MQ Mobile integration
mq_gateway	IBM MQSeries integration
nms_training_simulator	Network Management System Training Simulator
partitioning	Information Life Cycle Management
powerflow	Power Flow, Optimization, FLM, Suggested Switching
scada	SCADA Extension
service_alert	Service Alert
sgg	Oracle Utilities Smart Grid Gateway integration
stormman	Storm Management
suggested_switching	Suggested Switching
switching	Switching Management
water	This is a water utility. Automatically translates electrical-specific text and disables electrical-specific applications.
volt_var	Network Optimization

Run Automated Setup

Oracle has an automated process that sets up the database schema and directory structure. Any scripts, SQL files, or data files that are properly set up, named and installed will automatically get picked up and used by this process. The automated setup process will use various SQL files mentioned in this section to build the initial data model.

nms-setup

The `nms-setup` script must be run on the model build host machine, which is the machine on which MBService is running. This process loads scripts, SQL, and data files that are properly configured and installed. The script makes liberal use of ISQL, which submits all SQL files to the database to be run. The syntax is:

```
nms-setup [[-clean [-noVerify] [-reset]] | [-offline]] [-showme]
[-o [logFile]] [-noInherit] [-debug] [-noMigrations] [-post]
[-migrationOnly] [-cust_schema] [-help]
```

The following table describes the `nms-setup` command line options.

Option Variable	Description
-clean	Destroys the current model in order to build a new model. A prompt requires the user to verify this option. After this, a rebuilt model will still retain and use the same internal device identifiers (handles). This is useful for continuity of reporting before and after a *clean* model build. This option runs the entire set of SQL files, including model schema, retain, and configuration schema SQL files that drop and recreate the full set of NMS tables.
-noVerify	Bypasses the interactive verification prompt that opens for the `-clean` option.
-reset	Resets the generation of internal device identifiers (handles). If `-reset` is used with `-clean`, a model built afterward will not be relatable to the previous model, even though they may look the same.
-offline	Preserves the data model, but erases the real-time and historic information concerning the model, such as tags, permits or notes. Configuration changes made directly to the database may be lost. This runs the retain and configuration schema SQL files to drop and recreate their NMS tables. Without this or the -clean option, retain SQL files will not be run.
-showme	Prints the complete list in sequential order of scripts, SQL, and data files that are loaded or executed during the model build. Child scripts are indented in the list to easily identify parents. This option must be included in the database table or directory creation scripts in order to work properly.
-o [logFile]	If the **-o** parameter is specified, output will go to the log with the specified logFile name, except if "**-o -**" is used, in which case output will go to stdout.
-debug	Turns on debug; does nothing.
-noMigrations	Skips the automatic PR migration process. Use this option with caution, as it deviates from the supported process.
-post	Run post-setup script at end of setup. This is automatic when run with `-clean` or `-offline`. This option should **not** be used in dual-environment configuration.
-migrationOnly	Only run model migrations - not configuration.
-cust_schema	Create the customer schema tables.
-help	Print this help.

nms-setup Log File

nsm-setup automatically sends its output to a log file in **$NMS_LOG_DIR**. The standard naming convention is:

```
setup.[date].[time].log
```

The log file named is amended when any combination of the -clean, -offline, or -showme parameters are used:

- `setup_clean.[date].[time].log`
- `setup_offline.[date].[time].log`
- `setup_showme.[date].[time].log`
- `setup_clean_showme.[date].[time].log`
- `setup_offline_showme.[date].[time].log`

When output is sent to a log file, a single line will be sent to the console indicating the name of the log file. The first line of the log file shows the arguments that were passed to **nsm-setup**.

nms-post-setup

The nms-post-setup script must be run after the `nms-setup` unless `nms-setup` was run with the `-clean`, `-offline`, or `-post` option (in which case `nms-setup` will have automatically run `nms-post-setup`). This process loads scripts, SQL, and data files which will adversely affect the active NMS environment in dual-environment configuration. Running nms-post-setup will make the current environment the active environment.

Option Variable	Description
-o [logFile]	If the `-o` parameter is specified, output will go to the log with the specified logFile name, except if `-o -` is used, in which case output will go to stdout.
-showme	Prints the complete list in sequential order of scripts, SQL, and data files that are loaded or executed during the model build. Child scripts are indented in the list to easily identify parents. This option must be included in the database table or directory creation scripts in order to work properly.
-noMigrations	Skips the automatic PR migration process. Use this option with caution, as it deviates from the supported process.
-help	Print this help.

nms-post-setup sends its output to a log file in $NMS_LOG_DIR named `setup_post.[date].[time].log`.

NMS_CONFIG_ORDER Variable

The NMS_CONFIG_ORDER variable indicates configurations to use. The setup process looks only for the files containing the values specified in this variable. The syntax is:

```
NMS_CONFIG_ORDER="[project] nms"
```

The first argument is the name of the customer or project. The last argument is the name of the default base configuration. There may be multiple configurations specified between the first and last arguments. When the system boots it processes the arguments from right to left, so it first loads the base configuration. Then it moves on to the previous argument and loads the associated configuration files if they exist. The process continues until each argument is processed.

The setup process runs a large set of shell and SQL scripts that set up all aspects of the NMS model. The right-most value of the NMS_CONFIG_ORDER environment variable identifies a *base*, or predefined configuration. By default, the setup process sets up the model in the predefined configuration. However, the setup script contains numerous "hooks" that when encountered, install project-specific configuration that overrides the base configuration.

For example, if project XYZ defines a base model stdbase, then the NMS_CONFIG_ORDER environment variable is set to "xyz stdbase". The stdbase configuration is used by default, with project-specific files overriding stdbase files when encountered. The stdbase configuration may contain a script (stdbase-mb-preprocessor) that sets up the data model for the stdbase version of the preprocessor. Project XYZ uses a different preprocessor with a different setup. The **NMS** setup process has a hook for a [project]-mb-preprocessor file, so any file of this form with the project prefix as specified by NMS_CONFIG_ORDER (in this case, xyz-mb-preprocessor) is called in place of the stdbase version. The exact details are dependent upon the nature of the "hook" involved. Some hooks are set up to call both the project script and the base script, while others will only call one or the other.

Project Specific Supplementary Setup Process

At the very end of the nms-setup process is a hook to add project specific additional setup processes using the optional [project]-supplement script. Typical processes added in this script include:

- setting the next available index on certain classes to a special/higher number
- setting initial high level control zone names/numbers
- setting up special lookup tables used for model build preprocessing
- preload ces_users and validations from an external system
- setup SCADA device/point mapping tables
- add custom triggers to the model for integration to other system

Linking In Customers

In order for Oracle Utilities Network Management System Trouble Management to run, customer information must be linked into the model. This information is assumed to be in the database, whether explicitly loaded or whether linked in as a synonym. Oracle requires that the table that contains the customer information be joined to the SUPPLY_NODES table in the CES_CUSTOMERS table

Population of the CES_CUSTOMERS Table

The CES_CUSTOMERS table is populated with details about customers, their meters, and their locations. It includes information from the following tables:

- CU_CUSTOMERS
- CU_SERVICE_LOCATIONS
- CU_METERS
- CU_SERVICE_POINTS
- CU_DERS

To update the Oracle Utilities Network Management System customer model, project-specific customer import processes will drop and rebuild mirror versions of these tables, named:

- CU_CUSTOMERS_CIS
- CU_SERVICE_LOCATIONS_CIS
- CU_METERS_CIS
- CU_SERVICE_POINTS_CIS
- CU_DERS_CIS

They will then run `nms-update-customers`, which will perform change detection between the CU_*_CIS tables and their Oracle Utilities Network Management System counterparts, perform incremental updates to them, and re-create the CES_CUSTOMERS table.

> **Note**: If you do not want to update the CU_* tables or you do not have an updated set of CU_*_CIS tables, you should add the "`-no_pre_process`" option to the call to `nms-update-customers` and the CU_* tables will remain unchanged.

From the CES_CUSTOMERS table, a smaller (summary) table must be extracted that is called CUSTOMER_SUM.

Population of the CUSTOMER_SUM Table

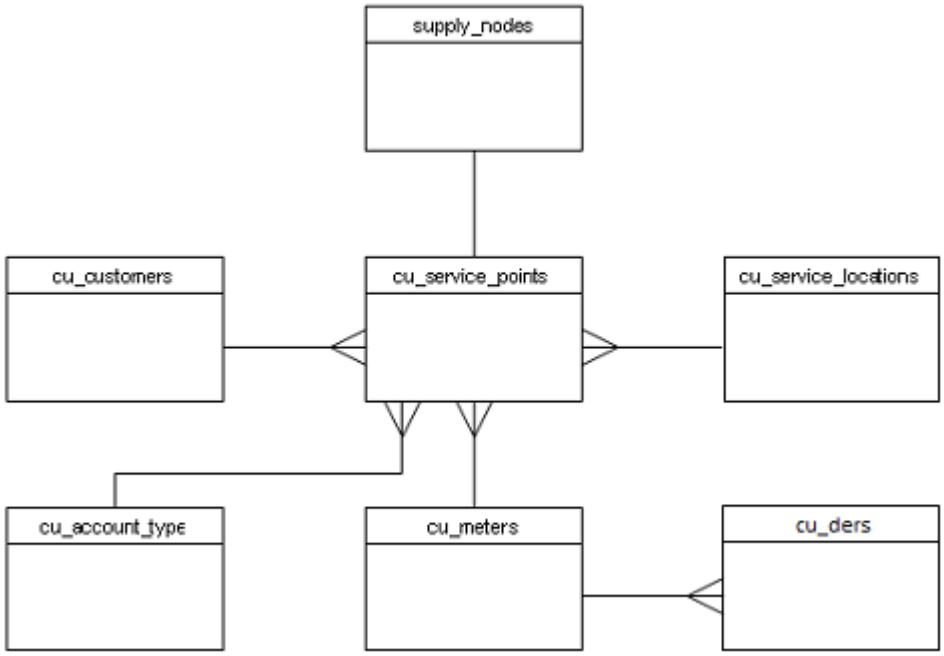
The CUSTOMER_SUM table is a smaller extraction of the CES_CUSTOMERS information in which the customer information is summarized. JMService uses this for faster calculations. Depending on the definition of CUSTOMER_SUM table, a fresh extraction may be required after each model build.

Customer Model - Logical Data Model

This section provides an overview of the logical view of the Oracle Utilities Network Management System customer model. Where the MultiSpeak data model uses Customer, Service Location and Meter entities, the Oracle Utilities Network Management System model adds the notion of a Service Point to increase flexibility, and provide for improved performance of the physical implementation. Additionally, the Oracle Utilities Network Management System model extends beyond the basic MultiSpeak model in the following ways:

- Supports more than one meter per service location.
- MultiSpeak attributes not required for NMS purposes are not required, such as billing information (acRecvBal, acRecvCur, …) and meterology information (kwh, multiplier, …)
- Provides model extensions to support important attributes not currently defined by MultiSpeak but necessary for NMS purposes.
- Supports customer-defined attributes for read-only purposes with no requirement for use in analysis.

This model, when joined with the Supply Node information in the Oracle Utilities Network Management System database (supply_nodes), results in the following E-R diagram:

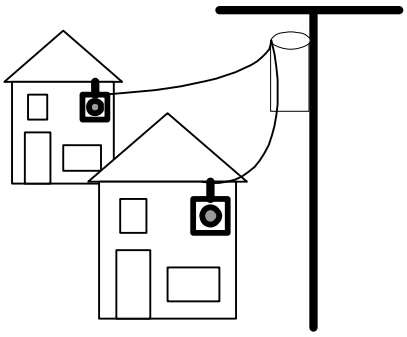


Residential Model

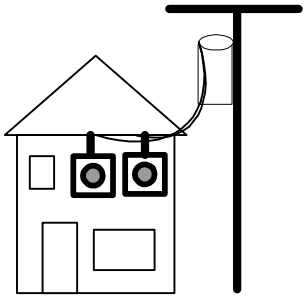
In this extended model, it is recognized that the occurrence of multiple meters is reasonably common, where each meter may have different rate codes associated.

Although the occurrence of multiple transformers is much less frequent than multiple meters, there are also several possible configurations of meters and transformers, with different electrical arrangements. Often, multiple transformers will occur on (geographically) large sites (for example, factory, airport, shopping mall, and so forth), where it is appropriate and helpful (from the perspective of outage analysis) to have multiple service locations defined for the site which aid in readily locating the appropriate transformer.

The following pictures depict some simple examples of the usage of this customer model. The first example shows two service locations, each with a meter connected to a distribution transformer.



The second example is an account with a single location with two meters, which is described through the definition of a customer account, a service location and two meters. The service location is associated with a distribution transformer.



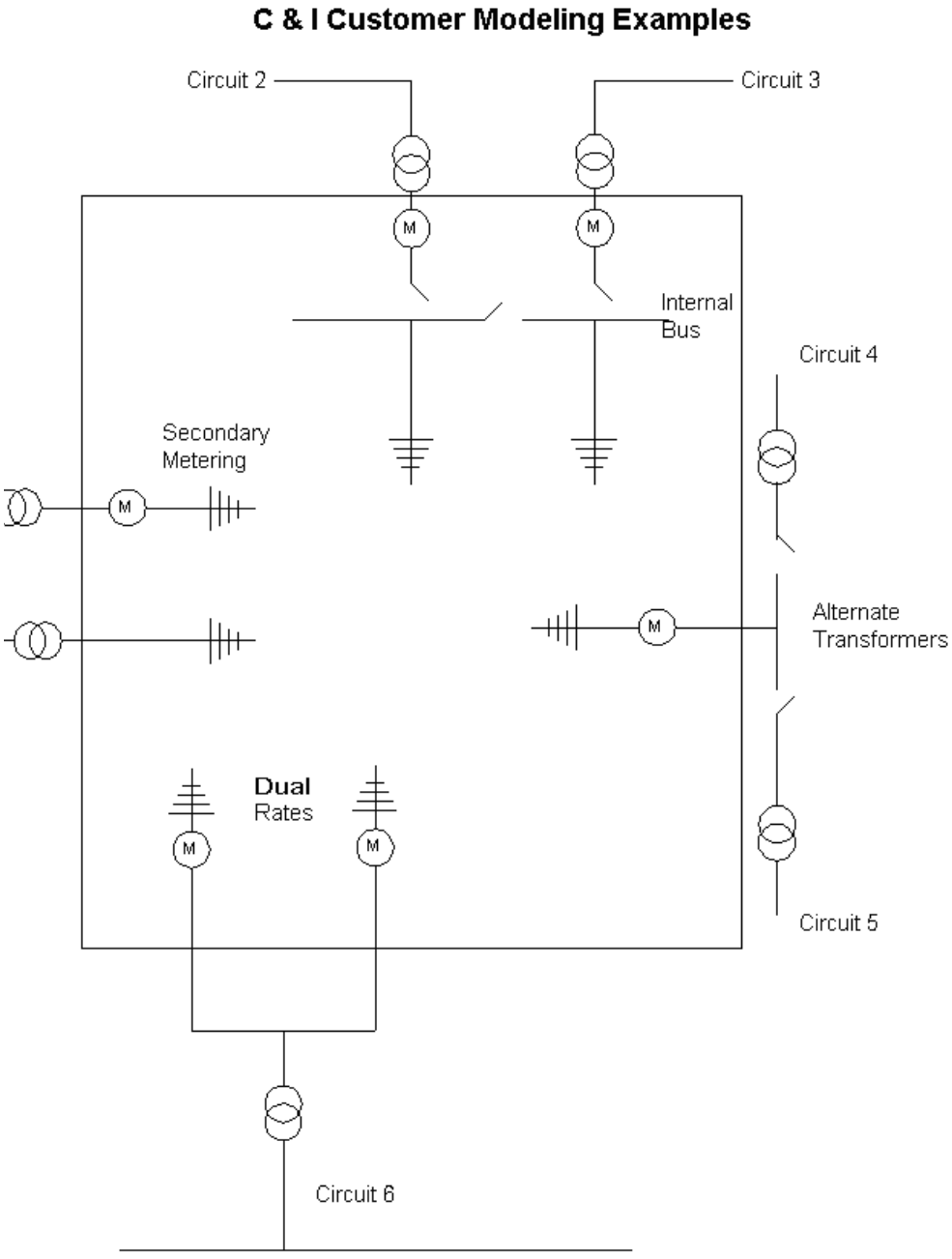
A third example would be a combination of the two previous examples, where a single customer account was responsible for the billing related to all of the above service locations. A more sophisticated example of residential metering is provided in the appendix.

Commercial and Industrial (C & I) Model

Many Commercial and Industrial situations are more complicated than residential metering. In these cases, a variety of configurations of meters, transformers and circuits must be addressed. The variations include:

- Primary metering, where the meter is placed on the high side of the transformer
- Internal buses, where two transformers can be used with two meters, feeding an internal bus
- Alternate transformers, where a meter can be switched to one of two transformers, each on a different circuit
- A single transformer feeding two meters, where different rates apply to each meter

The following diagram illustrates these examples.



Customer Model Database Schemas

The following section provides schema descriptions for the data and tables that are relevant to the Customer Model. It should be noted that the naming convention used internally is slightly different than the convention used in MultiSpeak or CIM exchange formats, due to the case-insensitive nature of Oracle RDBMS.

Customer Model Database Tables

The purpose of this section is to provide descriptions of the data and tables that support the implementation of the Oracle Utilities Network Management System customer model. These descriptions address only the data elements that are relevant to the customer model. The actual database tables may contain additional fields, but the other fields are not relevant to the customer model and are not described here.

Req Key Values	Meaning	Comment
N	Not required	Not needed for standard ces_customers table.
C	Configured in standard ces_customers table.	Not all columns referenced in the ces_customers table are required for a given implementation - inclusion of some columns can be project-specific.
Y	Required	Used in standard ces_customers table – still may not be 100% required. Actual requirements are generally project specific.

Customers Table

The **cu_customers** table is used to manage customer accounts. While the primary key is cust_id, this typically may have the same value as account_number.

Req	Column Name	Data Type	Description
Y,C	cust_id	NUMBER NOT NULL,	Primary key – may be generated.
N	cust_account_number	VARCHAR2(30) NOT NULL	Customer account number.
N	cust_billing_account	VARCHAR2(13) NULL	Customer billing account number.
Y,C	cust_name	VARCHAR2(90) NULL	Name of the customer; concatenation of last, first and middle names, or business name.
N	cust_last_name	VARCHAR2(30) NULL	Last name.
N	cust_first_name	VARCHAR2(30) NULL	First name. Typically, this is only populated for residential customers.
N	cust_middle_name	VARCHAR2(30) NULL	Middle name or initial.
Y,C	cust_home_ac	NUMBER(3) NULL	Phone area code for the home phone.
Y,C	cust_home_phone	NUMBER(7) NULL	Phone number for the home phone.
N	cust_day_ac	NUMBER(3) NULL	Phone area code for the work phone.
N	cust_day_phone	NUMBER(7) NULL	Phone number for the work phone.
N	cust_day_phone_ex	NUMBER(7) NULL	Typically, day phone numbers are related to customers' work phone numbers, which generally include extensions.
N	cust_bill_addr_1	VARCHAR2(50) NULL	Street address of the billing address. Note that billing address fields are usually populated only if different from the address held in the cu_service_point table.
N	cust_bill_addr_2	VARCHAR2(50) NULL	Second line, if necessary, of street address of the billing address.
N	cust_bill_addr_3	VARCHAR2(50) NULL	Third line, if necessary, of street address of the billing address.
N	cust_bill_addr_4	VARCHAR2(50) NULL	Fourth line, if necessary, of street address of the billing address.
N	cust_bill_city	VARCHAR2(30) NULL,	City of the billing address.
N	cust_bill_state	VARCHAR2(30) NULL	State of the billing address.
N	cust_bill_postcode_1	VARCHAR2(10) NULL	First 5 zip code numbers for US.
N	cust_bill_postcode_2	VARCHAR2(10) NULL	Second 4 zip code numbers for US.
C	cust_name_initials	VARCHAR2(3) NULL	The customer initials. Possibly used for certain soundex type searching if a customer wants to enable it - not often. Not necessary.

Req	Column Name	Data Type	Description
N	cust_comment	VARCHAR2(255)NULL	General field provided to support additional information about the customer, such as 30ft ladder, assult-case, crit-pmp-station, etc.
N	cust_user_def_1	VARCHAR2(255) NULL	These user-defined fields support the inclusion of other desired data not covered in the core fields. These fields can be extracted for project specific reporting.
N	cust_user_def_2	VARCHAR2(255) NULL	
N	cust_user_def_3	VARCHAR2(255) NULL	
N	cust_user_def_4	VARCHAR2(255) NULL	
N	last_update_time	DATE	Time of last update for record. Generally, set internally when a record is updated - not via external CIS.
Y	birth	DATE	Time the record was activated
Y	death	DATE	Time the record was deactivated
Y	to_die	VARCHAR(1)	(Internal use.) Used to flag whether the record is being given a death time.

Service Locations Table

The purpose of the cu_service_locations table is to manage locations (premises) at which a customer is served. A customer account may have multiple service locations.

Req	Column Name	Data Type	Description
Y,C	serv_loc_id	NUMBER NOT NULL	Primary key – may be generated.
N	serv_type	VARCHAR2(2) NULL	The type of service at this location. (electrical or gas). Only necessary for utilities that support multiple service types.
N	serv_status	VARCHAR2(50) NULL	Electrical service status of the service location. For example: INA – Inactive ACT – Active PDI – Pending Disconnect Can be used to coordinate business processes around how to handle customer disconnects (for example, update the day before). Each project needs to discuss these.
Y,C	serv_account_number	VARCHAR2(30) NOT NULL	The service account number which will be used for call entry purposes, and the account number used in createIncident XML.
Y,C	serv_revenue_class	VARCHAR2(30) NULL	Revenue class for the service location.
N	serv_load_mgmt	NUMBER NULL	Binary - whether or not there is load management at this Service Location
Y,C	serv_concat_address	VARCHAR2(200) NULL	Concatenated address of the service address 1, 2, 3, and 4.
N	serv_special_needs	VARCHAR2(1) NULL	Identifies any special needs of the customer.
N	serv_priority	VARCHAR2(32) NULL	Mapped to ces_customers.priority. This defines the meaningful customer type value the utility uses internally. This value will be displayed on troubleInfo as well.
N	serv_addr_1	VARCHAR2(50) NULL	First line of street address of the service address.
N	serv_addr_2	VARCHAR2(50) NULL	Second line, if necessary, of street address of the service address.
N	serv_addr_3	VARCHAR2(50) NULL	Third line, if necessary, of street address of the service address.
N	serv_addr_4	VARCHAR2 (50) NULL	Third line, if necessary, of street address of the service address.
N	serv_city	VARCHAR2(25) NULL	City of the service location.
N	serv_state	VARCHAR2(25) NULL	State of the service location.
Y,C	serv_city_state	VARCHAR2(50) NULL	This field contains the data that will appear in the ces_customers.CITY_STATE field.
Y,C	serv_postcode_1	VARCHAR2(10) NULL	First 5 Zip Code numbers for US.
N	serv_postcode_2	VARCHAR2(10) NULL	Second 4 Zip Code numbers for US.
N	serv_user_geog_1	VARCHAR2(25) NULL	User geo codes typically used for political areas, such as counties, tax districts, etc.
N	serv_user_geog_2	VARCHAR2(25) NULL	
Y,C	serv_town	VARCHAR2(3) NULL	The town or county for the customer.
Y,C	serv_str_block	VARCHAR2(20) NULL	Block number - used in searches.

Req	Column Name	Data Type	Description
N	serv_str_pfix	VARCHAR2(10) NULL	The 'R' in R 321 Rolling Rd (R rear, F front, A adjacent, etc.)
Y,C	serv_str_struc	VARCHAR2(20) NULL	Structure relates to apartments, units, piers, docks, warehouse, slip, etc.
N	serv_str_name	VARCHAR2(30) NULL	Name of the street (Main Street).
N	serv_str_cdl_dir	VARCHAR2(10) NULL	Cardinal direction (N, S, E, W).
N	serv_str_sfix	VARCHAR2(10) NULL	ST, PKY, PLC, DR, RD, AVE, etc.
Y,C	serv_lot	VARCHAR2(10) NULL	Lot number – used in searches.
Y,C	serv_apt	VARCHAR2(8) NULL	Apartment number.
N	serv_elec_addr	VARCHAR2(50) NULL	Elec address used in searches.
N	serv_sic	VARCHAR2(8) NULL	Standard Industrial Code.
N	serv_comment	VARCHAR2(255) NULL	General comment about the service location.
Y,C	serv_cumulative_priority	NUMBER NULL	Summation of priority codes for this location.
Y,C	serv_life_support	NUMBER NULL	Indicates if this is a life-support customer.
Y,C	serv_a_priority	NUMBER NULL	**A** customer defined flag, 0 or 1
Y,C	serv_b_priority	NUMBER NULL	**B** customer defined flag, 0 or 1
Y,C	serv_c_priority	NUMBER NULL	**C** customer defined flag, 0 or 1 – often emergency customers.
Y,C	serv_d_priority	NUMBER NULL	**D** customer defined flag, 0 or 1 – often medical customers.
Y,C	serv_e_priority	NUMBER NULL	**E** customer defined flag, 0 or 1 – often entertainment customers.
Y,C	serv_f_priority	NUMBER NULL	**F** customer defined flag, 0 or 1 – often fire customers.
Y,C	serv_g_priority	NUMBER NULL	**G** customer defined flag, 0 or 1 – often government customers.
Y,C	serv_h_priority	NUMBER NULL	**H** customer defined flag, 0 or 1 – often hospital customers.
Y,C	serv_i_priority	NUMBER NULL	**I** customer defined flag, 0 or 1
Y,C	serv_j_priority	NUMBER NULL	**J** customer defined flag, 0 or 1
Y,C	serv_k_priority	NUMBER NULL	**K** customer defined flag, 0 or 1 – often keycustomers.
Y,C	serv_l_priority	NUMBER NULL	**L** customer defined flag, 0 or 1 – often life support customers.
Y,C	serv_m_priority	NUMBER NULL	**M** customer defined flag, 0 or 1 – often manufacturing customers.
Y,C	serv_n_priority	NUMBER NULL	**N** customer defined flag, 0 or 1 – often nursing home customers.
Y,C	serv_o_priority	NUMBER NULL	**O** customer defined flag, 0 or 1
Y,C	serv_p_priority	NUMBER NULL	**P** customer defined flag, 0 or 1 – often police customers.
Y,C	serv_q_priority	NUMBER NULL	**Q** customer defined flag, 0 or 1
Y,C	serv_r_priority	NUMBER NULL	**R** customer defined flag, 0 or 1

Req	Column Name	Data Type	Description
Y,C	serv_s_priority	NUMBER NULL	**S** customer defined flag, 0 or 1 – often sensitive customers.
Y,C	serv_t_priority	NUMBER NULL	**T** customer defined flag, 0 or 1 – often transportation customers.
Y,C	serv_u_priority	NUMBER NULL	**U** customer defined flag, 0 or 1
Y,C	serv_v_priority	NUMBER NULL	**V** customer defined flag, 0 or 1
Y,C	serv_w_priority	NUMBER NULL	**W** customer defined flag, 0 or 1
Y,C	serv_x_priority	NUMBER NULL	**X** customer defined flag, 0 or 1
Y,C	serv_y_priority	NUMBER NULL	**Y** customer defined flag, 0 or 1
Y,C	serv_z_priority	NUMBER NULL	**Z** customer defined flag, 0 or 1 – often 911 customers.
Y,C	serv_map_loc_x	NUMBER NULL	GPS lat/long or other mapping coordinates.
Y,C	serv_map_loc_y	NUMBER NULL	
N	serv_user_def_1	VARCHAR2(255) NULL	These user-defined fields support other desired data not covered in the core fields. These fields can be extracted for project-specific reporting purposes.
N	serv_user_def_2	VARCHAR2(255) NULL	
N	serv_user_def_3	VARCHAR2(255) NULL	
N	serv_user_def_4	VARCHAR2(255) NULL	
N	last_update_time	DATE	Time of last update for record.
Y	birth	DATE	Time the record was activated
Y	death	DATE	Time the record was deactivated
Y	to_die	VARCHAR(1)	(Internal use.) Used to flag whether the record is being given a death time.

Meters Table

The cu_meters table describes meters that might exist at a service location. The use of meters is optional (but increasingly common) within Oracle Utilities Network Management System. Meter information is required for a project which intends to utilize integration with an Automated Meter Reading Infrastructure.

The cu_service_points table tracks the relationship between a meter (cu_meters) and a customer account (cu_customers) and service location (cu_service_locations).

Req	Column Name	Data Type	Description
Y,C	meter_id	NUMBER NOT NULL	Primary key – may be generated.
Y,C	meter_no	VARCHAR2(20) NOT NULL	Meter number.
N	meter_serial_number	VARCHAR2(20) NULL	Serial number on the meter.
N	meter_type	VARCHAR2(20) NULL	Type of meter (gas, electric, water, etc.).
N	meter_manufacturer	VARCHAR2(20) NULL	Manufacturer of the meter.
N	meter_phases	VARCHAR2(1) NULL	Phase(s) connected to the meter (IE 1, 2, or 3).
N	meter_rate_code	VARCHAR2(65) NULL	Rate code for the meter.
N	meter_user_def_1	VARCHAR2(255) NULL	These user-defined fields support other desired data not covered in the core fields.These fields can be extracted for project-specific reporting purposes.
N	meter_user_def_2	VARCHAR2(255) NULL	
N	meter_user_def_3	VARCHAR2(255) NULL	
N	meter_user_def_4	VARCHAR2(255) NULL	
N	last_update_time	DATE	Time of last update for record.
Y	birth	DATE	Time the record was activated
Y	death	DATE	Time the record was deactivated
Y	to_die	VARCHAR(1)	(Internal use.) Used to flag whether the record is being given a death time.

Account Type Table

The purpose of the cu_account_type table is to contain a configuration of the valid Account Types that can be specified for a Service Point record. The initial loading of customer data populates this table. There is often only one row in this table (for electrical service).

cu_account_type		
Column Name	**Data Type**	**Description**
acctyp_account_type	VARCHAR2(10) NOT NULL	Electric, Gas, Propane, Appliance Repair, etc.
acctyp_user_def_1	VARCHAR2(255) NULL	These user-defined fields support other desired data not covered in the core fields. These fields can be extracted for project-specific reporting purposes.
acctyp_user_def_2	VARCHAR2(255) NULL	
acctyp_user_def_3	VARCHAR2(255) NULL	
acctyp_user_def_4	VARCHAR2(255) NULL	

Service Points Table

The purpose of the cu_service_points table is to manage the linkages between the cu_customers, cu_service_locations, cu_meters, cu_account_type and supply_nodes tables.

Key indexes are placed on this table for performance. History can be tracked, by setting active_fl to 'N' to identify that a record is now historical. No timestamp is used to track when a service point went out of service and the cu_service_points table is not intended nor recommended as a long term repository for service point history.

Req	Column Name	Data Type	Description
Y,C	serv_point_id	VARCHAR2(64) NOT NULL	Primary key. If the CIS cannot provide a unique value, use a generated key (for example, by combining cust_id, serv_loc_id and meter_id columns). This is used for CIS-to-NMS integration. For Customer Care & Billing (CC&B) integration in Oracle Utilities Network Management System 1.10, this is the CC&B Service Point Id. (See below for related info on ces_customers).
Y,C	cust_id	NUMBER NOT NULL	Foreign key ref to the cu_customers table.
Y,C	serv_loc_id	NUMBER NOT NULL	Foreign key ref to the cu_service_locations table.
Y,C	meter_id	NUMBER NOT NULL	Foreign key ref to the cu_meters table.
Y,C	device_id	VARCHAR2(25) NOT NULL	Foreign key ref to the supply_nodes table. This field is critical and necessary, as it ties Oracle Utilities Network Management System to the CIS.
N	feeder_id	VARCHAR2(10) NULL	Foreign key ref to the supply nodes table. Note this field is non-critical and generally not necessary.
Y,C	active_fl	VARCHAR2(1) NOT NULL	Identifies currently active records. Generally, this is always ‘Y’ as there is little provision or need for inactive records in the system. Inactive records are generally removed from this table.
N	create_dttm	DATE NOT NULL,	Timestamp for the record's creation.

Req	Column Name	Data Type	Description
Y,C	account_type	VARCHAR2(10) NOT NULL	Foreign key to the cu_account_type table.
N	last_update_time	DATE	Time of last update.
Y	birth	DATE	Time the record was activated
Y	death	DATE	Time the record was deactivated
Y	to_die	VARCHAR(1)	(Internal use.) Used to flag whether the record is being given a death time.

DERs Table

The purpose of the cu_ders table is to manage the distributed energy resources (DERs) that may be connected to a meter. The use of this table is optional depending on whether distributed energy resources are present on the associated distribution network. The information stored within this table can be used within the Trouble Info Customer List to display relevant information related to DERs associated to a particular customer. If distributed energy resources do not exist within the GIS data, this table can be used by the model preprocessing to artificially inject these devices into the NMS model. The preprocessing can determine the service transformer associated to the DER and then add a DER of the indicated technology type to the NMS model. .

Req	Column Name	Data Type	Description
Y,C	der_id	NUMBER NOT NULL	Primary key – may be generated.
Y,C	der_name	VARCHAR2(80) NOT NULL	Name of the DER.
Y,C	meter_id	NUMBER NOT NULL	The corresponding ID of the connected meter.
N	der_type	VARCHAR2(30) NULL	Technology type for the DER (PV, wind, etc.).
N	der_phases	VARCHAR2(1) NULL	DER phasing.
N	rated_size	VARCHAR2(30) NULL	The size of the DER in kVA.
N	nominal_voltage	NUMBER NULL	The nominal voltage value of the DER.
N	der_user_def_1	VARCHAR2(255) NULL	These user-defined fields support other desired data not covered in the core fields.These fields can be extracted for project-specific reporting purposes.
N	der_user_def_2	VARCHAR2(255) NULL	
N	der_user_def_3	VARCHAR2(255) NULL	
N	der_user_def_4	VARCHAR2(255) NULL	
N	last_update_time	DATE	Time of last update for record.
Y	birth	DATE	Time the record was activated
Y	death	DATE	Time the record was deactivated
Y	to_die	VARCHAR(1)	(Internal use.) Used to flag whether the record is being given a death time.

Linkages to Other Tables

The customer model has linkages to other tables in the Oracle Utilities Network Management System model. The primary linkage between utility customers and the Oracle Utilities Network Management System electrical network model is the *device_id* column. The definitive table linkage is between supply_nodes.device_id and cu_service_points.device_id. From the perspective of the cu_service_points table, the device_id field is used to uniquely identify the electrical network model element (supply node) which supplies power to a service point (customer).

In general, an Oracle Utilities Network Management System *supply node* is any place on the model where a utility customer can be connected to receive electrical power. For customers that wish to model secondary network, this supply point can be associated with a single customer/meter. For customers that are only interested in modeling primary distribution circuits, the supply node is often associated with a secondary transformer.

The Oracle Utilities Network Management System electrical data model is implemented under the assumption that the source for the electrical network model data (generally a Geographic Information System) and the source for the utility customer data (generally a Customer Information System) understand and maintain this customer-to-supply-node relationship. The accuracy of this linkage is critical for reliable trouble call handling and outage reporting. Without this linkage, customer trouble calls enter the system as fuzzy calls and outage reports have diminished accuracy.

Customer Model Views

The purpose of this section is to describe the views that support existing Oracle Utilities Network Management System software, and provide compatibility for this customer model with existing installations.

CES Customers View

The ces_customers view (or table) is derived from the cu_customers, cu_service_locations, cu_meters, cu_service_points, cu_ders, and supply_nodes tables. It provides a flat customer view that is utilized by various Oracle Utilities Network Management System services and applications such as JMService, Web Call Entry and others.

Note: all of the following fields are required.

Displayed Column Name	Originating Table	Column in originating table
id	cu_service_points	serv_point_id
h_cls	supply_nodes	device_cls
h_idx	supply_nodes	device_idx
supply_idx	supply_nodes	h_idx
meter_number	cu_meters	meter_no
device_id	supply_nodes	device_id
account_type	cu_service_points	account_type
account_number (not null)	cu_service_locations	serv_account_number
account_name	cu_customers	cust_name
address_building	cu_service_locations	serv_str_struc
block	cu_service_locations	serv_str_block
address	cu_service_locations	serv_concat_address
city_state	cu_service_locations	serv_city_state
zip_code	cu_service_locations	serv_postcode_1
phone_area	cu_customers	cust_day_ac
phone_number	cu_customers	cust_day_phone
priority	cu_service_locations	serv_cumulative_priority
a_priority	cu_service_locations	serv_a_priority
b_priority	cu_service_locations	serv_b_priority
c_priority	cu_service_locations	serv_c_priority
d_priority	cu_service_locations	serv_d_priority
e_priority	cu_service_locations	serv_e_priority
f_priority	cu_service_locations	serv_f_priority
g_priority	cu_service_locations	serv_g_priority
h_priority	cu_service_locations	serv_h_priority

Displayed Column Name	Originating Table	Column in originating table
i_priority	cu_service_locations	serv_i_priority
j_priority	cu_service_locations	serv_j_priority
k_priority	cu_service_locations	serv_k_priority
l_priority	cu_service_locations	serv_l_priority
m_priority	cu_service_locations	serv_m_priority
n_priority	cu_service_locations	serv_n_priority
o_priority	cu_service_locations	serv_o_priority
p_priority	cu_service_locations	serv_p_priority
q_priority	cu_service_locations	serv_q_priority
r_priority	cu_service_locations	serv_r_priority
s_priority	cu_service_locations	serv_s_priority
t_priority	cu_service_locations	serv_t_priority
u_priority	cu_service_locations	serv_u_priority
v_priority	cu_service_locations	serv_v_priority
w_priority	cu_service_locations	serv_w_priority
x_priority	cu_service_locations	serv_x_priority
y_priority	cu_service_locations	serv_y_priority
z_priority	cu_service_locations	serv_z_priority
life_support	cu_service_locations	serv_life_support
avg_revenue	cu_service_locations	serv_revenue_class
name_initials	cu_customers	cust_name_initials
town	cu_service_locations	serv_town
feeder_id	supply_nodes	feeder_id
lot	cu_service_locations	serv_lot
apt	cu_service_locations	serv_apt
cust_id (not null)	cu_customers	cust_id
meter_id (not null)	cu_meters	meter_id
serv_loc_id (not null)	cu_service_locations	serv_loc_id
amr_enabled	cu_meters	amr_enabled
der_unit_count	cu_ders	count of records for the meter_id
x_coord	cu_service_locations	serv_map_loc_x
y_coord	cu_service_locations	serv_map_loc_y
user_geographic_log	cu_service_locations	serv_user_geog_1

CES_CUSTOMERS History

The CES_CUSTOMERS table only holds active records, so the CES_CUSTOMERS_HISTORY was created to hold historical CES_CUSTOMERS records.

When first created, active CES_CUSTOMERS records will be copied to CES_CUSTOMERS_HISTORY and back-dated so that their birth values are January 1, 2000. This is to match potential historical outages for these records. A new death column is added as null for these records.

As the CES_CUSTOMERS changes, the active CES_CUSTOMERS_HISTORY record is set with a death date, and a new record is added with the current date.

NMS_ACCOUNTS_HISTORY

Table NMS_ACCOUNTS_HISTORY maintains history of customer account information in NMS. This information is not used by NMS itself but can be useful for analytics tools.

The NMS_ACCOUNTS_HISTORY table always contains the following columns:

ACCOUNT_NUMBER: Account number. Populated from CES_CUSTOMERS_HISTORY.ACCOUNT_NUMBER

BIRTH: Date/time when this record became active

DEATH: Date/time when this record became inactive (special value '1/1/4000' indicates that the record is currently active)

PTN_DATE: Date/time used for partitioning

TO_DIE: Column used during incremental update procedure

Additional columns can be added to the NMS_ACCOUNTS_HISTORY table through configuration parameters in the CES_PARAMETERS table.

APP: 'BI'

ATTRIB: ACCT_HIST_COLUMN' followed by arbitrary string making the value unique

VALUE: Name of the column of the CES_CUSTOMERS_HISTORY table. Column with the same name will be added to the NMS_ACCOUNTS_HISTORY table.

Example

```
INSERT INTO CES_PARAMETERS (APP, ATTRIB, VALUE, SITE) VALUES
  ('BI', 'ACCT_HIST_COLUMN1', 'ACCOUNT_NAME', 'all');
INSERT INTO CES_PARAMETERS (APP, ATTRIB, VALUE, SITE) VALUES
  ('BI', 'ACCT_HIST_COLUMN2', 'ACCOUNT_TYPE', 'all');
INSERT INTO CES_PARAMETERS (APP, ATTRIB, VALUE, SITE) VALUES
  ('BI', 'ACCT_HIST_COLUMN3', 'SERV_LOC_ID', 'all');
INSERT INTO CES_PARAMETERS (APP, ATTRIB, VALUE, SITE) VALUES
  ('BI', 'ACCT_HIST_COLUMN4', 'AVG_REVENUE', 'all');
```

If column configuration is changed, then the NMS_ACCOUNTS_HISTORY table needs to be dropped. Next execution of the nms-update-customers command will recreate and repopulate the table with the new set of columns.

Customer Sum View

Within Oracle Utilities Network Management System, the customer_sum view (or table) is used primarily by JMService to identify the number of customers, critical customers, etc. on each supply node. The customer_sum view/table is typically generated from the ces_customers table/view. It is simply a summation of the customer model and is designed to provide more efficient outage impact estimates.

Note: all of the following fields are required.

Displayed Column Name	Originating Table	Column in originating table
supply_cls	supply_nodes	h_cls (=994)
supply_idx	supply_nodes	h_idx
device_id	supply_nodes	device_id
revenue	cu_service_locations	serv_revenue_class
customer_count	count(distinct cu_service_points)	cust_id
critical_a	sum(cu_service_locations)	serv_a_priority
critical_b	sum(cu_service_locations)	serv_b_priority
critical_c	sum(cu_service_locations)	serv_c_priority
critical_d	sum(cu_service_locations)	serv_d_priority
critical_e	sum(cu_service_locations)	serv_e_priority
critical_f	sum(cu_service_locations)	serv_f_priority
critical_g	sum(cu_service_locations)	serv_g_priority
critical_h	sum(cu_service_locations)	serv_h_priority
critical_i	sum(cu_service_locations)	serv_i_priority
critical_j	sum(cu_service_locations)	serv_j_priority
critical_k	sum(cu_service_locations)	serv_k_priority
critical_l	sum(cu_service_locations)	serv_l_priority
critical_m	sum(cu_service_locations)	serv_m_priority
critical_n	sum(cu_service_locations)	serv_n_priority
critical_o	sum(cu_service_locations)	serv_o_priority
critical_p	sum(cu_service_locations)	serv_p_priority
critical_q	sum(cu_service_locations)	serv_q_priority
critical_r	sum(cu_service_locations)	serv_r_priority
critical_s	sum(cu_service_locations)	serv_s_priority
critical_t	sum(cu_service_locations)	serv_t_priority
critical_u	sum(cu_service_locations)	serv_u_priority
critical_v	sum(cu_service_locations)	serv_v_priority
critical_w	sum(cu_service_locations)	serv_w_priority

critical_x	sum(cu_service_locations)	serv_x_priority
critical_y	sum(cu_service_locations)	serv_y_priority
critical_z	sum(cu_service_locations)	serv_z_priority
critical_both	sum(cu_service_locations)	Sum of all critical customers of any types.
x_coord	point_coordinates	x_coord
y_coord	point_coordinates	y_coord
ddo		Historical – likely should be removed at some point. Often set the same as customer_count to satisfy JMService.
zip_code	ces_customers	zip_code
city_state	ces_customers	city_state
user_geographic_loc	ces_customers	user_geographic_loc

Model Build Process

Model Build with a Preprocessor

In most cases, customers will place source data files into a designated directory and run the `nms-model-build` script. This script takes no arguments and builds whatever maps are recognized by the `[project]-maps-to-build` script. When the build completes, any completed maps will have import files automatically placed in a designated directory. In some cases, models may be built directly from import files.

Customer Model Build Scripts

The following table describes the model build scripts.

Script	Description

nms-model-build	Builds the maps recognized by `[project]-maps-to-build`. Upon completion, the `${OPERATIONS_MODELS}/patches/done` directory contains import files for the built maps.
	
	The script will check for a `[project]-model-build` script, if it exists, it will be called instead of running the rest of the `nms-model-build` script.
	
	Options
	`-noprebuild`
	`-nopostbuild`
	
	The parameters will cause the `nms-model-build` script to skip prebuild script execution or postbuild script execution.

[project]-model-build	If a `[project]-model-build` script exists, it will be called by `nms-model-build` and will be used as the project configured script to run the model build process. Most projects will not use this script.
[project]-build-map	Required for any model build process that has a model preprocessor. Takes a map name and generates an import file for that map. The resulting import file is placed in the `$OPERATIONS_MODELS/patches` directory. This script should preserve the `.mp` file time stamp if altered by any of its processing of the map data. See the `OPAL-build-map` as an example.
nms-build-maps	This script takes multiple map prefixes as parameters. Any maps supplied will be built as a single model transaction. This is recommended when there is a model transaction involving multiple maps, especially if facilities are being transferred from one map to another. This script checks that there are no conflicting pending maps in separate work orders. Since this check takes a few seconds, it can be bypassed by setting the CES_PARAMETERS entry for 'NMS', 'CHECK_MAP_DEPENDENCIES' to 'false'. This does not run any `-prebuild` or `-postbuild` scripts.

Script	Description
[project]-maps-to-build	Required for all model build processes. Identifies and prints a list (single line, space separated) of all maps that are queued up to be built. Model `.mb` files in the patches directory should be included in the list of maps to build including the `.mb` extension. All other maps to build should be reported without extensions. The resulting list of maps may include the same map twice, once for a new map to be preprocessed and once for a map waiting to be applied from the patches directory. Use the `OPAL-maps-to-build` as an example.
[project]-postbuild	Although not a required element of the model build, project-specific needs may call for an additional process after each model build. The additional process is carried out by the `[project]-postbuild` script. It is run after the `nms-model-build` script builds a complete set of maps. Common reasons for this process include recalculations of control zones, a fresh extraction of the CUSTOMER_SUM table, or a recache of DDService.
[project]-prebuild	Although not a required element of the model build, project-specific needs may call for an additional process before each model build. The `[project]-prebuild` script carries out the additional process. It is run before the `nms-model-build` script builds a complete set of maps. This process is rarely needed.

Model Build with a Post-Processor

If a post-processor is needed for the model build, you should create and install the [project]-postbuild script. If the post-processor requires patches to be applied to the model, it will build import files and put them in the patches directory. The nms-build-map script can be called with the `-noVerify` option to build each patch without user interaction.

Constructing the Model

To ensure correct model construction, complete these steps:

1. If a model build preprocessor is being used, make sure that the expected scripts are created and installed. These are `[project]-build-map` and `[project]-maps-to-build`.
2. When new files are brought to the system, place them in the appropriate directory on the master server before initiating the model build.
   - Import files should go into the `${OPERATIONS_MODELS}/patches` directory.
   - Preprocessor input files will probably go into a project-specific directory. An example of a commonly used directory is `${OPERATIONS_MODELS}/mp`.
3. Log into the master server as the administrative user and initiate a model build by typing:

   ```
   nms-model-build
   ```

   Or, if you want to produce a build log, enter the following:

```
nms-model-build | tee model_build.$(date "+%Y%m%d.%H%M%S").log 2>&1
```

Result: Each import file will be processed, updating the Operating Model and Graphic Presentation files.

4. Wait for the user prompt before continuing further model build operations. This process may take some time.
5. Review the error output information contained in the errors directory.

The Model Build Preprocessor

Oracle Utilities Network Management System obtains descriptions of the physical, electrical, and topological infrastructure from CAD, GIS and AM/FM systems through the model builder and associated preprocessors. The purpose of a preprocessor is to extract information from a source (GIS, CAD, AM/FM, etc.) and convert it to the neutral Oracle Utilities Network Management System import (.mb) format. From this format, it is processed by the model builder to determine and apply actual changes to the Oracle Utilities Network Management System operations model.

When the product is to be configured for a customer, there is a need to populate the corresponding Operations Model. Typically customers will have data stored within one or more forms: within a GIS, within a CAD product, in an RDBMS, or in flat files.

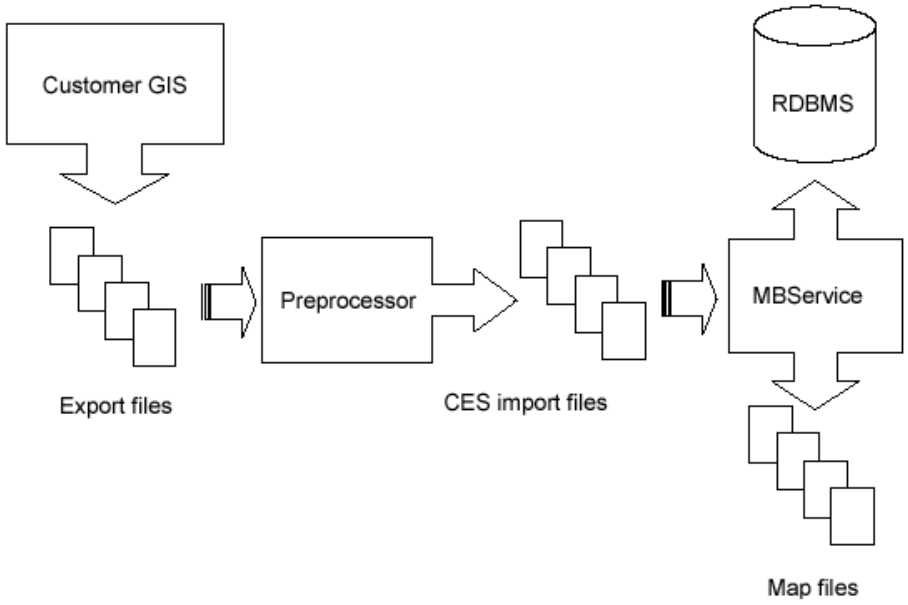
The information within these forms can either be directly extracted or preprocessed to a form which can be presented to the Model Build interface.

Model Build Basics

Model Build is a process of steps that will generate an operational topological representation of client's existing GIS. A single segment of data (partition) passes through four stages during its incorporation into the Operations Model:

- Extraction
- Preprocessing
- Model Build (MB Service)
- Completed Operations Model

The following figure provides an overview of the model build process:



Extraction

The graphical representations of objects that will be modeled, along with the associated attributes, are grouped and exported into external files in a format that the preprocessor is capable of reading. It is at this stage that the partitioning of the model into geographic grids or schematic diagrams is typically determined. These partition file names must match the partition name, as identified within the file, with the requisite file extension.

Preprocessing

The preprocessor reads the files generated by the extraction process and constructs an Import file which models the extracted portion. The preprocessor tends to be a major development task, taking weeks or months to complete.

Model Build

The Model Build (or MB Service) parses the Import file, verifies basic model consistency, applies the contained changes to the Operations Model Database, and commits the changes as part of the final model.

Completed Operations Model

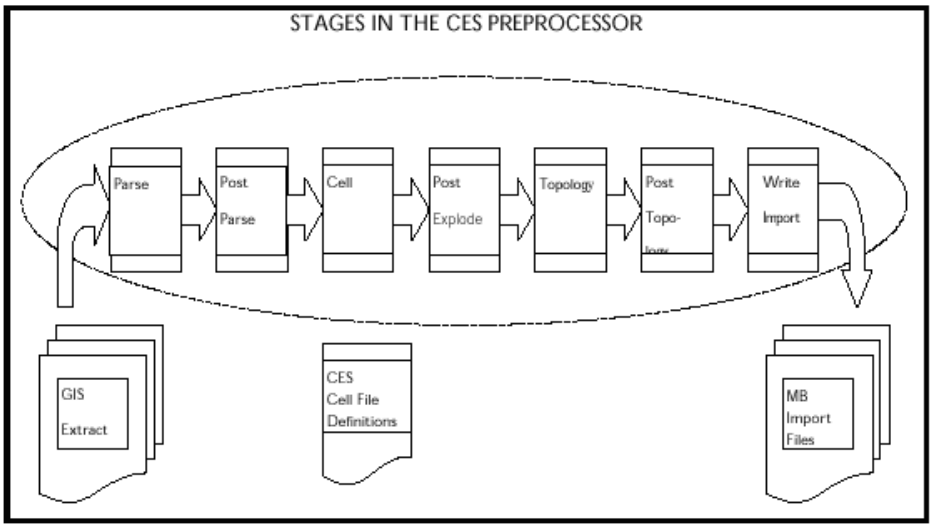
The completed model consists of new or updated partitions and new or revised entries within the core model database schema.

Model Preprocessor

The preprocessor reads (parses) the files generated by the extraction process and constructs an import file that accurately models the extracted portion. The end result of completing a preprocessor is a script that is capable of accepting customer source GIS data files and generating import files.

The Model Preprocessor can be broken into individual stages called: Parse, Post Parse, Cell Explosion, Post Explode, Topology Construction, Post Topology, and Model Build Import file generation.

The following figure illustrates the stages in the preprocessor:



Parse Stage

The Parser reads the client GIS model from external files created by the Extraction process into a data structure known as an Entity Set. After this phase is completed, the resulting Entity Set will be a 'skeleton' for the complete model. The activities completed in this stage are not client specific; it will be more specific to a standard data file format (e.g., AutoCAD's DXF format, Intergraph's ISFF format, etc.). Each individual graphical object (e.g., point, line, or text) will be represented in an output file.

- **Post Parse**: Client specific processing that is used to accommodate any modification of the data that may be required prior to Cell Explosion.
- **Cell Explosion**: Cell explosion is the central phase of preprocessing. It is here that the conversion of the raw graphical objects to model objects is accomplished. The graphical objects are mapped to objects, which will appear in client's final model.
- **Post Explode**: Allows for client specific processing after Cell Explosion.
- **Topology Construction**: The inter-device connectivity for all electrical objects is constructed in this stage. The connectivity can either be explicit (i.e. 'To' and 'From' node identifiers) or based on proximity.
- **Post Topology**: The final opportunity for client specific processing.
- **Model Build Import File Generation**

Cell Explosion

The central phase of preprocessing is the conversion of graphical objects into full-fledged model objects; this conversion from a graphical object to a model object can involve a wide range of operations. These operations are specified in a text file [client]_devices.cel, which is called the explosion definition file.

The operations that may be accomplished during this phase include the following:

- **Handle Assignment -** This requires that a graphical entity be mapped to a particular class of model objects (for example, switch, transformer, device annotation, road, water boundary, etc.) and that an index number, unique within that class, be assigned to this object.
- **Attribute Manipulation** - Attributes can be added, removed or renamed. They can also be assigned new values based upon combinations of other attribute values or the result of mathematical calculations.
- **Expansion/Replacement of One Object by Multiple Objects** - For example a transformer in the mapping system could be exploded into a transformer with a switch and a network protector.
- **Creation of Aggregate Objects** - One object may be used to represent a group of objects. For example, a recloser object may in fact represent the recloser along with a by-pass switch, a load switch, and a source switch. All of these component objects may be created and bundled into a single aggregate object during this phase.
- **Elimination of Un-Necessary Objects** - Any object not explicitly 'matched' during this phase will be eliminated; thus, this stage acts as a filter.
- **Assignment of Core Properties** - For example, phase, nominal status, NCG, and symbology can be assigned as default values for all devices.
- **Daughter Object Creation** - Creating new entities based upon information taken from an existing object.
- **Classification of Objects as Background** – Sets the location of an object to a background partition.
- **Diagnostic Messaging** – Aids in debugging or as a method to configure customer specific error messages with customer defined attributes.

Model objects have handles (class and index), attributes and aliases, geometry, and optionally aggregate object specification, all of which are supported through the explosion preprocessor.

To understand the cell definitions, which specify how an object is recognized and processed during cell explosion, one should understand two fundamental ideas:

1. "Parent" and "daughter" objects
2. String expansion.

Parent and Daughter Objects

Those objects, which enter the cell explosion process from the parser (or the post-parse processing) and which are recognized (or matched) by a definition, are considered to be "parent" objects (or, at least, potential parents); any new graphic objects created by the cell definition which matched the parent are considered "daughter" objects.

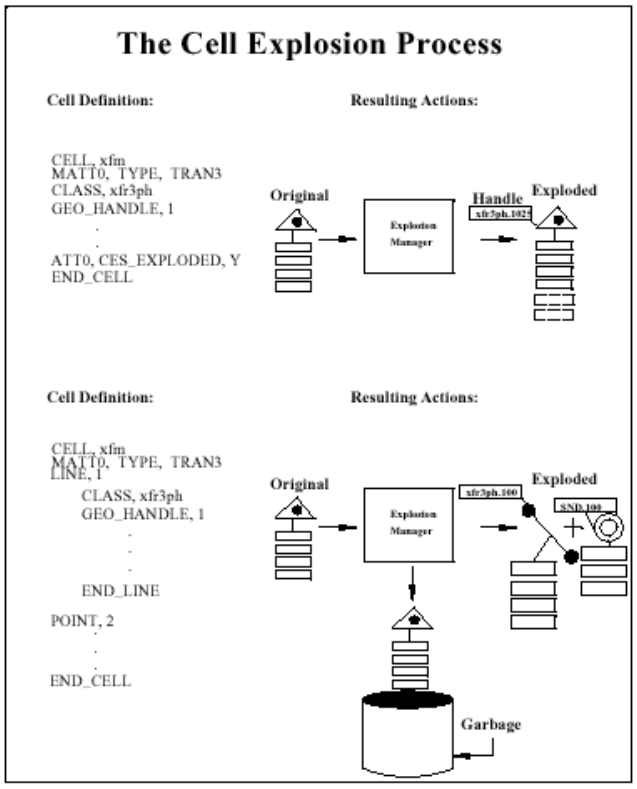
There are 4 outcomes for an object after cell explosion:

1. The parent object may pass through and be modified by cell explosion without giving rise to daughter objects.
2. The parent object may pass through cell explosion while giving rise to one or more daughter objects.
3. The parent object may be eliminated by cell explosion yet give rise to daughter objects, which survive and proceed to the succeeding stages.
4. The parent object may be eliminated by cell explosion and not give rise to daughter objects.

> **Note**: Any object that has an attribute named "CES_EXPLODED" with a value of "Y" will pass through this process; all other objects are eliminated.

Commonly, if the parent gives rise to daughter objects, the parent dies, but transfers some of its attributes to the resulting daughters through use of the ATT keyword.

The following illustration depicts outcomes 1 and 2 for an object:



String Expansion

When assigning new attributes, you may want the values for these new attributes to be formed from existing attributes, either by simply copying an existing value, or by combining and/or transforming the old values. This process is accomplished by "string expansion" which replaces or expands an attribute name into the full string representing that attribute's value. In cell definitions, enclosing an attribute name in square brackets indicates that you intend for this attribute name to be expanded; for example, the form "[FEEDER_ID]" will be replaced by the value of the FEEDER_ID attribute, such as "6992" (assuming that such an attribute exists for the matched object). In addition to this simple expansion, there are several specialized forms of string expansion that can be summarized as follows:

1. **Substring**

   - **Delimiter Based**

     Indicated by "<"or ">". This this form returns the substring before or after the first occurrence of the delimiting character. The delimiting character is the character immediately following the "<" or the ">".

     For example, if TAG= "XYZ.553", then [<.[TAG]] returns the substring preceding the first period (".") in the TAG attribute value, in this case, "XYZ". Likewise, [>.[TAG]] returns the substring following the period, which would be "553".

   Note: When nesting a simple expansion form (e.g., [TAG]) within a delimiter based expansion form; you can discard the inner square brackets. Thus, "[<.TAG]" is equivalent to "[<.[TAG]]".

   - **Position Based**

     Indicated by "@" -- this form returns the substring beginning and ending at the given character positions.

     Using the example from above where TAG="XYZ.553", the notation [@(1:2)[TAG]] extracts the substring from the value of the TAG attribute, which begins with character position 1 (position 0 being the first character) and ends with character position 2. In other words, it extracts a two-character substring, beginning from the second position, returning the value "YZ".

   Note: The character position can be specified relative to the end of the string by using the "$" character to represent the last position in the string. For example, "[@($-1:$)[TAG]]" returns the last two characters "53". Also note that a single character can be extracted by specifying the start and end positions as the same character, e.g., "[@(2:2)[TAG]]" returns the third character, "Z".

2. **Codelist**

   These can be used to map or convert an input value into the corresponding output value.

   - **Basic Lookup Table:**

     To create the "lookup table", we use the CODE keyword. The format for the table is:

     ```
     CODE, [listname] ,input value, outputvalue.
     ```

     For example:

     ```
     CODE, RANK_LIST, E, 1
     CODE, RANK_LIST, R, 2
     CODE, RANK_LIST, P, 4
     ```

Creates a lookup table with three entries or mappings.

(A default code, returned when the given input value is not in the table, can be defined for a list using the DEFAULT_CODE keyword; for example, DEFAULT_CODE, RANK_LIST, 1 means that any input value other than E, R or P results in the output of a "1".)

To actually look up or convert a value, we use the codelist form of string expansion, indicated by a "%".

```
[%RANK_LIST.[RANK_CODE]]
```

Will return "1" if the RANK_CODE attribute is "E"; "2" if the RANK_CODE is "R"; and "4" if the RANK_CODE is "P".

- Database Lookup:

  This works the same as the basic lookup table but the entries are stored in a database table. There are 2 formats for database lookups:

  - DBCODE

    The table name (which also serves as the list name), the input column name, and the output column name are defined using the DBCODE keyword. The format for the DBCODE is:

    ```
    DBCODE, [tablename], [input column], [output column]
    ```

    For example:

    ```
    DBCODE, feeder_ncg, feeder_name, ncg_id
    ```

    Means that there exists a database table called "feeder_ncg" which has an input value column called "feeder_name" and an output value column "ncg_id".

  - NAMED_DBCODE

    NAMED_DBCODE is similar to DBCODE except it takes a list name that is different from the table name. It is used in cases where there is a need for 2 codelists based on the same database table but with different input and output columns. The format is:

    ```
    NAMED_DBCODE, [listname], [tablename], [input column],
    [output column>]
    ```

  A default code, returned when the given input value is not in the table, can be defined for a list using the DEFAULT_CODE keyword. For example, DEFAULT_CODE, feeder_ncg, 1 means that any input value other than what has been defined results in the output of a "1". Additionally, a special DEFAULT_CODE value can be assigned with the value specified as "--INTEGER_SEQUENTIAL--".

  For example:

  ```
  DBCODE, feeder_ncg, feeder_name, ncg_id
  DEFAULT_CODE, feeder_ncg, --INTEGER_SEQUENTIAL--
  ```

  Means if a lookup into the table named feeder_ncg does not have a match, the default action will be to select the maximum value of ncg_ids in the table, add one to the ncg_id, and create a new record with the given feeder name and the incremental maximum ncg_id.

  Accessing the table is the same as the basic lookup table mentioned above.

- **Math Functions**

  Mathematical functions can be calculated by using the input value to access a "pseudo-codelist." "List name" has one of the following values:

  MATH_SIN

  MATH_COS

  MATH_TAN

  MATH_ASIN

  MATH_ACOS

  MATH_ATAN

  MATH_LOG

  MATH_LOG10

  MATH_EXP

  MATH_SQRT

  MATH_CEIL(round up to next greatest number)

  MATH_FLOOR(round down to next lowest number)

  MATH_FABS(absolute value; for example, -4.5 becomes 4.5)

  MATH_RPN(math function in reverse polish notation)

  For example, to calculate the sine of an ANGLE attribute:

  ```
  [%MATH_SIN.[ANGLE]]
  ```

- **Coordinate Lookup**:

  The coordinates of an object can be accessed using a form that mimics a codelist lookup:

  [%COORDINATE.FIRSTX] returns the first X coordinate of the object

  [%COORDINATE.LASTX] returns the last X coordinate of the object

  [%COORDINATE.FIRSTY] returns the first Y coordinate of the object

  [%COORDINATE.LASTY] returns the last Y coordinate of the object

3. Default Value

A default value can be specified which will be returned if the result of string expansion would otherwise be an empty string. This is indicated by enclosing a default value between two caret symbols ("^").

For example: "[^PRIMARY^[PRI_CIRCUIT_ID]]" returns a value of "PRIMARY" in any case where the PRI_CIRCUIT_ID attribute is non-existent or empty.

If a default value is not specified, then a "String Expansion Error" message will occur.

4. Special Attributes

Some properties of an object can be accessed as if they were attributes by using one of the special names given below, preceded by a double dollar sign:

CLS(cell number)

IDX(index number)

X1(1st or primary X coordinate)

Y1(1st or primary Y coordinate)

Xn(subsequent X coordinate)

Yn(subsequent Y coordinate)

COORD_CNT(number of coordinates)

MAP_CLASS(class number of partition)

MAP_NAME(full name of partition)

CELL_NAME(cell name, *i.e.*, the set of instructions for an object)

CLS_NAME(actual name of class rather than number)

For example, [$$CELL_NAME] returns the name of the "cell" within the cell definition file that was matched by the current object.

5. **Handle Reference**

   One daughter object can access the class and index number of another daughter object by using the following two forms:

```
$<#>.CLS
$<#>.IDX
```

   For example, in daughter object #2, the class number of daughter #1 can be accessed by the form: "[$1.CLS]" and index number of daughter #1 can be accessed by the form: "[$1.IDX]".

   Note: A common practical application of this form of string expansion is to assign the DEVICE_CLS and DEVICE_IDX attributes of a SND attached to its corresponding transformer.

Available Cell Explosion Keywords

This section provides descriptions, syntax, and examples for available cell explosion keywords.

Global (outside all cell definitions)

- **CODE** - Defines an entry in a code conversion lookup table. See String Expansion Section.
- **DBCODE** - Defines an entry in a database table. See String Expansion Section.
- **DEFAULT_CODE** - Sets default values for codelist. See String Expansion Section.
- **INCLUDE** - Reads definitions from another file.

  INCLUDE, <name of file to include>

```
INCLUDE,  /users/xyz/data/xyz_devices.cel
```

- **NAMED_DBCODE** - Allows for definitions of more than one codelist from a single database table.
- **TEMPLATE** - Uses a template definition.
- **USE** - Sets default values for an entity's properties.

  USE, [KEYWORD], [value]

```
USE, PHASE, abc
```

Shared (used by both parent objects & daughter graphic objects)

- **ATT[n]** - Sets the value of an attribute. There is no limit on the number of ATT[n] records that can exist in the cell definitions. [n] is currently a placeholder, usually set to 0 (zero).

  ATT[n], [att_name], [att_value]

  ```
  ATT0, feeder, [@(9:12) [ACAD_layer]]
  ATT0, riser, N
  ```

- **ATTR_INDEX** - The string that follows this keyword will be used to assign an index unique for an object of this object's class; usually, the string will be formed by expansion of one or more attributes.

  ATTR_INDEX, [n]

  ```
  ATTR_INDEX,
  ```

- **BND_HANDLE** - Indicates that the index for this object should be provided by the boundary-node handle manager.

  ```
  BND_HANDLE, 1
  ```

- **CLASS** - Sets the class of object to explicit value.

  CLASS, <class name>

  ```
  CLASS, Xfm
  ```

- **DATT[n]** - Dynamic attribute name. [n] is currently a placeholder, usually set to 0 (zero).

  DATT[n], <att_name>, <att_value>

  ```
  DATT1,  [%LOCATION.[^0^[WITHIN_SITE_IPID]]],~
  4901.[^0^[WITHIN_SITE_IPID]]
  ```

- **GEO_HANDLE** - Indicates that a unique index should be generated based upon the object's class and geographical coordinates.

  ```
  GEO_HANDLE, 1
  ```

- **INDEX** - Sets the index of object to an explicit value.

  INDEX, <n>

  ```
  INDEX, 533
  ```

- **MARK_BGD** - Marks an object as background and sets its location to the background partition.

  ```
  MARK_BGD, 1
  ```

- **MSG[n] (or MESSAGE[n])** - Prints a message to standard output when this definition is used, where [n] is either 0, 1, 2, or 3.

  MSG[1|2|3], <message text>
  MESSAGE[1|2|3], <message text>

  ```
  MSG1, Warning:  Found stray fuse
  MSG2, Handle:  [$$CLS] . [$$IDX]
  MSG3, At (X,Y) of ([$$X1],[$$Y1])
  ```

- **NCG** - Set the entity's Network Control Group (NCG) property. (Program-style preprocessor only)

  NCG, <n>

  ```
  NCG, [@(9:12)[ACAD_layer]]
  ```

- **NOMINAL_STATE** - Sets the entity's 'NOMINAL_STATE' property. The value can be an integer typically between 0 and 15 or the key words OPEN or CLOSED.

  NOMINAL_STATE, [n]|OPEN|CLOSED

  ```
  NOMINAL_STATE, CLOSED
  ```

- **OPT_ATT[n]** - Sets an optional value of an attribute. Will not report a string error message if the value fails on attribute expansion. [n] is currently a placeholder, usually set to 0 (zero).

  OPT_ATT[n], [att_name], [att_value]

  ```
  OPT_ATT1, From_Node_Bnd, [NODE1_BND]
  ```

- **OPT_DATT[n] -** Sets an optional dynamic attribute name. Will not report a string error message if the attribute name fails on attribute expansion. [n] is currently a placeholder, usually set to 0 (zero).

  OPT_DATT[n]

  ```
  OPT_DATT1, [%LOCATION.[^0^[WITHIN_SITE_IPID]]],~
      4901.[^0^[WITHIN_SITE_IPID]]
  ```

- **PHASE** - Sets the entity's 'PHASE' property (e.g., to ABC).

  PHASE, [n]

  ```
  PHASE, [%PHASE_LIST.[@(6:8)[ACAD_layer]]]
  ```

- **STRING** - Sets the value of the text string for this entity. (TEXT objects only)

  STRING, [string]

  ```
  STRING, [kVAr]
  ```

- **SUB_BND** - Indicates that this object is a substation boundary node and that its index should be assigned based upon the supplied string (usually the feeder or circuit identifier).

  ```
  SUB_BND, 1
  ```

- **SYM_ID** - Sets the symbology-state-class to an explicit value, rather than its default value, which is the same as the class number.

  SYM_ID, [n]

  ```
  SYM_ID, 1304
  ```

- **VOLTS** - Sets the entity's 'VOLTS' property. (Program-style preprocessor only)

  VOLTS, [n]

  ```
  VOLTS, [voltage] 1000 *
  ```

Parent Object ("explosionDef") Only

- **AGGREGATE/_ POINT/_LINE/_TEXT** - Creates a graphic object of the specified kind that becomes a component of the overall aggregate device. AGGREGATE and AGGREGATE_LINE require 2 coordinates; AGGREGATE_POINT and AGGREGATE_TEXT require one coordinate. All AGGREGATE definition types require an END_AGGREGATE. (Obsolete)

  AGGREGATE, [n]

  ```
  AGGREGATE, 4
  ```

  AGGREGATE_ POINT, [n]

  ```
  AGGREGATE_POINT, 1
  ```

  AGGREGATE_ LINE, [n]

```
AGGREGATE_ LINE, 3
```

AGGREGATE_ TEXT, [n]

```
AGGREGATE_ TEXT, 2
```

- **CELL** - Begins the definition for one device type. The cell definition file can contain many sets of cell definitions. All CELL definitions require an END_CELL.

  CELL, [name]

  ```
  CELL, uxfm2
  ```

- **END_CELL** - Ends an explosion definition.

  ```
  END_CELL
  ```

- **END_AGGREGATE** - Ends an aggregate definition.

  ```
  END_AGGREGATE
  ```

- **END_TEMP** – Ends a template definition.

  ```
  END_TEMP
  ```

- **MATT[n]** - Matching attribute of the object to explode. [n] is currently a placeholder, usually set to 0 (zero). There is no limit on the number of MATT[n] records a cell explosion definition may have, but for the explosion to occur, all must match.

  MATT[n], [attribute name], [target attribute value]

  ```
  MATTO, ACAD_objectType, INSERT
  ```

- **POINT/LINE/TEXT** - Creates a "daughter" graphic object of the specified kind. All POINT/LINE/TEXT definitions require an END_POINT/LINE/TEXT.

  POINT, [n]

  ```
  POINT, 3
  ```

  LINE, [n]

  ```
  LINE, 1
  ```

  TEXT, [n]

  ```
  TEXT, 5
  ```

- **POINT/LINE/TEXT WHEN [condition]** - Creates a "daughter" graphic object of the specified kind when the given condition is met. All POINT/LINE/TEXT definitions require an END_POINT/LINE/TEXT.

  POINT WHEN [condition]

  ```
  POINT WHEN
  ```

  LINE WHEN [condition]

  ```
  LINE WHEN
  ```

  TEXT WHEN [condition]

  ```
  TEXT WHEN
  ```

- **POINT/LINE/TEXT FO**R [variable] IN [List of Values] - Creates zero, one or multiple graphic objects of the specified kind, one object for each value in the supplied list. Use [variable] within the definition as if it were an attribute name. A special variable called "$$ICOUNT" can also be used to retrieve the number of the iteration. All POINT/LINE/TEXT definitions require an END_POINT/LINE/TEXT.

  POINT FOR [variable] IN [list of values]

  ```
  POINT FOR
  ```

  LINE FOR [variable] IN [list of values]

LINE FOR

TEXT FOR [variable] IN [list of values]

TEXT FOR

- **POINT/LINE/TEXT FOR [num-value] TIMES** - Creates zero, one or multiple graphic objects of the specified kind; number of objects specified by *<num-value*s>. ($$ICOUNT can be used just as for the previous form). All POINT/LINE/TEXT definitions require an END_POINT/LINE/TEXT.

  POINT FOR [numeric value] TIMES

  POINT FOR

  LINE FOR [numeric value] TIMES

  LINE FOR

  TEXT FOR [numeric value] TIMES

  TEXT FOR

- **REQUEST_HANDLE** - Indicates that the existing handle of this object should be replaced with one supplied by the Explosion manager's "ExplodeHandle" class. (Primarily for ISFF)

  ```
  REQUEST_HANDLE, 1
  ```

- **RMV[n]** - Removes an attribute. [n] is currently a placeholder, usually set to 0 (zero).

  RMV[n]

  ```
  RMV0, voltage
  ```

- **RNA[n]** - Renames an attribute. [n] is currently a placeholder, usually set to 0 (zero).

  RNA[n], [att_name], [new_att_name]

  ```
  RNA0, amp_content, amp_cont
  ```

- **TEXT_SCALE** - Specifies the scale factor for text. Used to allow the height of base text symbol to be used as a multiplier to the cell definition specified coordinates.

  TEXT_SCALE, [n]

  ```
  TEXT_SCALE, 1
  ```

  for example with the TEXT_SCALE, 1 specified and the base text object has a specified height of 400 and the COORD1, 10, 30 is specified, the resulting coordinates will be 400x10, 400x30 or 4000, 12000.

- **USE_REFERENCE** - Indicates that the index for this object should be based upon its corresponding reference object. (ISFF only) (Obsolete). For example:

  ```
  USE_REFERENCE, 1
  ```

  causes the FRAMME RB_REFPRMRY and RB_REFSCNDRY linkages to be used instead of the normal RB_PRIMRY and RB_SECNDRY.

Component "Daughter" Object ("explosionGrObject") Only

- **ABSOLUTE_COORDS** - Indicates that coordinate values are specified in absolute, "real-world" numbers; this over-rides the default behavior which is for numbers used in COORD statements to be taken as relative to the insertion point of the parent object (i.e. this insertion point corresponds to COORD 0.0, 0.0).

  ```
  ABSOLUTE_COORDS, 1
  ```

- **ANGLE** - Sets the text rotation for this entity. Horizontal is zero and the angle proceeds counter clockwise. (TEXT objects only)

ANGLE, [a]

```
ANGLE, 90
```

- **COORD/COORD[n]**- Sets relative/absolute coordinate of an object/endpoint.

  COORD, [x], [y]

  ```
  COORD, 1.0, 2.5
  ```

  COORD[n], [x], [y]

  ```
  COORD1, 0.0, 1.0
  COORD2, 1.0, 2.0
  ```

- **COMPONENT[n]** - Sets the aggregate sequence number and cell component number for a single component in the aggregate.

  COMPONENT[n], [agg_seq_num], [cell_comp_num]

  ```
  COMPONENT1, 1, 2
  ```

- **END_AGGREGATE** - Ends the definition of component graphic object.

  ```
  END_AGGREGATE
  ```

- **HEIGHT** - Sets the text height for this entity. (TEXT objects only)

  HEIGHT, <h>

  ```
  HEIGHT, 2
  ```

- **H_ORIENTATION** - Sets the horizontal justification of text. Values can be LEFT, CENTER, or RIGHT, or 0, 1, or 2. Default is LEFT. (TEXT objects only)

  H_ORIENTATION, [n]|LEFT|CENTER|RIGHT

  ```
  H_ORIENTATION, LEFT
  ```

- **USE_ROTATION** - Indicates that the rotation property of the original entity should be used to set the rotation for the component graphic object.

  ```
  USE_ROTATION, 1
  ```

- **V_ORIENTATION** - Sets the vertical justification of text. Values can be TOP, CENTER, or BOTTOM, or 0,1, or 2. Default is BOTTOM. (TEXT objects only)

  V_ORIENTATION, [n]|TOP|CENTER|BOTTOM

  ```
  V_ORIENTATION, 2
  ```

Special Attributes Set by Explode an Processed by mat2entityset.(script-pre-processor):

- **Alias -** Sets an alias for an attribute (both script- and program-style preprocessors)**.**

  ATT[n], ALIAS[dbtype], [value]

  ```
  ATT0, ALIAS[OPS], [LOC_NUM]
  ```

- **Diagram-id** - Sets the Diagram Id.

  ATT[n], DIAGRAM_ID, [value]

  ```
  ATT1, DIAGRAM_ID, [IPID]
  ```

- **Group –** Sets the Group code**.**

  ATT[n], CES_PP_GROUP|GROUP|Group|group, [value]

- **Local**

  ATT[n], LOCAL|Local|local, [value]

- **Locations** (not to be confused with LOCATIONS)

  ATTN[n],CES_LOCATION, [value]

  ```
  ATT1,  CES_LOCATION, 4901.[MID]
  ```

  ATTN[n], CES_LOCATION_DEFINITION, [value]

  ```
  ATT1,  CES_LOCATION_DEFINITION, 4901.[MID]
  ```

  ATT[n], CES_LOCATION_NAME, [value]

  ```
  ATT1,  CES_LOCATION_NAME, Pole [^?^[SUPPORT_NO]]
  ```

  ATT[n], CES_LOCATION_DESC, [value]

  ```
  ATT1,  CES_LOCATION_DESC, Pole defined by support/switch:~
     [^?^[SUPPORT_NO]]/[^?^[SWITCH_NAME]]
  ```

  ATT[n], CES_LOCATION_REFERENCE, [value]

  ```
  ATT1,  CES_LOCATION_REFERENCE, [%COORDINATE.FIRSTX],~
  [%COORDINATE.FIRSTY]
  ```

Network Control Group

ATTN[n], NCG|Ncg|ncg, [value]

```
ATT1,  NCG, [%feeder_ncg.[^UNKNOWN^[DISTRICT]]_ ~
[%ncg_volt.[^UNKNOWN^[VOLT_LEV]]]]
```

- **Rank**

  ATT[n], RANK|Rank|rank, [value]

  ```
  ATT1,  RANK, [%MATH_RPN.[%RANKU.[^NO^[URBAN]]]~
  [%RANKLC.[^UNKNOWN^[LINE_CATEGORY]]] + ~
  [%RANKV.[^0^[VOLT_LEV]] [^0^[VOLT_LEV]]] + ~
  [%RANKB11.[^0^[VOLT_LEV]] [^UNKNOWN^[DISTRICT]]] + ~
  [%RANKP.[^RYB^[PHASING]]] +]
  ```

- **Physical Property**

  ATT[n], CES_PHYS_PROP|PHYS_PROP|Phys_Prop|phys_prop|physical_property, [value]

  ```
  ATT0, CES_PHYS_PROP, [%MATH_RPN.[%PHYS_PROP.BACKBONE] [%PHYS_PROP.~
  [^OH^[OH_UG]]] +]
  ```

- **Topology specific**

  ATT[n], From_Node, [value]

  ```
  ATT1,  From_Node, [FROM_NODE]
  ```

  ATT[n], To_Node, [value]

  ```
  ATT1,  To_Node, [TO_NODE]
  ```

  ATT[n], Unique_id, [value]

  ```
  ATT1,  Unique_Id, [FROM_NODE]_[TO_NODE]_FID
  ```

- **Transition**

  ATT[n], TRANSITION_ID|Transition_ID|Transition_Id|transition_id, [value]

  ```
   ATT1,  TRANSITION_ID, 120
  ```

- **Voltage**

  ATT[n], VOLTAGE|Voltage|voltage, [value]

  ```
  ATT1,  VOLTAGE, [%VOLTS.[^UNKNOWN^[OPERATING_VOLTAGE]]]
  ```

Format for the Explosion Definition File

Devices are recognized, or 'matched', and appropriate manipulations are made based upon the descriptions or definitions contained in an explosion definition text file.

The general format for a single cell definition is as follows:

```
CELL, [cell-name]
[match-criteria]
[ [parent-object-actions] ]
[ [daughter-object-actions] ]
END_CELL
```

Remember, any object that has an attribute named "CES_EXPLODED" with a value of "Y" will pass through the explosion process (ATTO, CES_EXPLODE, Y); all other objects are eliminated.

Syntax

Cell Definition

1. One statement per line (the ~ can be used to continue on more than one line).
2. Comments begin with # and must be on a line by themselves.
3. Lines begin with keywords (always upper case).
4. Commas separate keywords and values.

   Value fields can be:

   - Attribute substituted using the syntax [<att name>] where the value of the <att name> for the currently exploded object will be substituted in the value string. See the examples in the line definition above.
   - Math functions in Reverse Polish Notation (RPN) with space delimitation. The keywords which support RPN automatically are:
     - ANGLE
     - HEIGHT
     - H_ORIENTATION
     - INDEX
     - NCG
     - NOMINAL_STATE
     - SYMBOLOGY
     - VOLTS
     - V_ORIENTATION

     For example, the following will be valid:

     ```
     COORD, 100.0, 300.0
     COORD, 100.0 [X_OFFSET] +, 300.0 [Y_OFFSET] +
     COORD, [X_OFFSET], [Y_OFFSET]
     ```

   Math operators supported include +, -, *, /, % (modulus) and ^ (exponentiation).

During the Parse phase of the preprocessor, the customer's raw data files are converted into an internal data structure known as an Entity Set wherein each individual graphical

object is represented by an Entity object. Each Entity object is read into the cell file and is processed separately. When creating a cell definition file, to decrease processing time:

1. Place filter cells at the top of the file. For example, cells with nothing but match criteria that will not be exploded.
2. Place cells with most abundant objects near the top of the file. For example, if a file contains 20 switches, 10,000 text objects and 500 transformers, place the text objects first, transformers next, and finally the switches.
3. Place most restrictive criteria cells for objects above general. Overhead transformers should be placed above generic transformers in the cell definition file.

Match Criteria

1. Use keyword MATT[n].
2. Basic form: MATT[n],<attribute name>,<target attribute value>.
3. Attribute name can be replaced by a string expansion.
4. Can use alternation of target values separated by |.

```
MATTO,[ACAD_layer],15kv-Bus|24kv-Bus|161kv-Bus
```

5. Multiple match criteria are logically "AND" ed together. All MATT[n] must return true before that cell will be used. For example, for the following cell to be used for an Entity object, all 3 lines must return true:

```
CELL, 01XF1
   MATTO, ACAD_objectType, INSERT
   MATTO, ACAD_blockName, 01XF1
   MATTO, [@(1:3) [ACAD_layer]], PRI
   …
```

Conditional Expressions

These have the form:

```
( (Boolean-Expression) ? true value | false value )ATT0, ALIAS[OPS],
( ([location]) ?  [location]|D:[ATTR] )
```

The supported syntax for Boolean expressions within cell-definition files is as follows:

```
<Expression> = <Expression> && <Expression>
   <Expression> || <Expression>
   !<Expression>
   (Expression)
   <String-Comparison>
   <Numeric-Comparison>
   <Term>
```

where

```
<String-Comparison> = <String> == <String>
   <String> != <String>
   <String> < <String>
   <String> > <String>
   <String> <= <String>
   <String> >= <String>
```

where

```
<Numeric-Comparison> = <Number> .eq. <Number>
   <Number> .ne. <Number>
```

```
<Number> .lt. <Number>
<Number> .le. <Number>
<Number> .gt. <Number>
<Number> .ge. <Number>
```

where

```
<Term> = <String> | <Number> | <Function-Call>
```

where

```
<String> = <Simple-String> | <Expand-Form>
```

where

`<Simple-String>` = double-quoted string of alphanumeric characters (*e.g.*, `"553"`).

`<Expand-Form>` = attribute or property name enclosed in square brackets (*e.g.*, [att_name])

`<Number>`

`<Function-Call>` = name of a standard function with argument(s) enclosed in matched parentheses.

Note: At present no standard functions have been implemented, so this feature should not be used.)

Operators are evaluated in the following order, with top most operators processed first. The operators used are:

```
!
< > <= > >= .lt. .gt. .le. .ge.
== != .eq. .ne.
&&
||
```

Examples:

```
([Layer] .eq. 501)
(([ObjectType] != "Primary Conductor") && ( [FeederId] .ne. 6800
))
( sin(Rotation) < 0.5 )
( ![UniqueId] )
```

Example of Cell Definitions

Transformer w/Supply Node

```
CELL, OverheadTransformer
  MATT0, CESMP_OBJ_CLASS, Transformer
  MATT0, [OhUg], OH
  MATT0, DIAGRAM_ID, Symbol

  LINE, 1
    ABSOLUTE_COORDS,1
    COORD1,[$$X1],[$$Y1]
    COORD2,[$$Xn],[$$Yn]

    # Definition attributes
    CLASS, xfm_oh
    SYM_ID, 2060[%phase_num.[^ABC^[Phase]]]
    ATTR_INDEX, [GUID]
    ATT0, ALIAS[OPS], [DeviceId]
    ATT0, ALIAS[GIS], [GisId]
    NCG, [%feeder_ncg.[CESMP_MAPNAME]]
    ATT0, NCG_FDR, [CESMP_MAPNAME]

    # Topology definition
    PHASE, [%phase_map.[^ABC^[Phase]]]
    NOMINAL_STATE, [%status_lookup.[^CLOSED^[NominalStatus]]]
    VOLTS, [%voltage.[^4160^[Voltage]]]
    PHY_PROPERTIES, [ces_physical_property]
    ATT0, From_Node, [_Connector0])
    ATT0, To_Node, [_Connector0]_SND

    RANK, [%phase_bit.[^ABC^[Phase]]]
 [%voltage_bit.[%voltage.[^4160^[Voltage]]]] +
    # Attribute mapping
    OPT_ATT0, facility_id,             [GisId]
    OPT_ATT0, device_name,             [DeviceId]
    OPT_ATT0, feeder_id_1,             [FeederName]
    OPT_ATT0, feeder_id_2,             [FeederName2]
    # Explode this object
    ATT0, CES_EXPLODED, Y
  END_LINE
    POINT, 6
      CLASS, SND
      ATTR_INDEX, [GUID]
      PHASE, [%phase_map.[Phase]]
      SYM_ID, 994
      NCG, [%feeder_ncg.[CESMP_MAPNAME]]
      COORD, 0, -1
      ATT0, Unique_Id, [_Connector0]_SND
      ATT0, device_cls, [$1.CLS]
      ATT0, device_idx, [$1.IDX]
      ATT0, device_id, [DeviceId]
      ATT0, feeder, [$$MAP_NAME]
      ATT0, phases, [%phase_num.[Phase]]
      ATT0, ncg, [%feeder_ncg.[CESMP_MAPNAME]]
      ATT0, CES_EXPLODED, Y
    END_POINT

END_CELL
```

Code Lookup Examples

This example shows how a lookup table can be used to convert the GIS phase to an NMS phase:

```
# CODE phase_map
#
CODE, phase_map, 1, A
CODE, phase_map, 2, B
CODE, phase_map, 4, C
CODE, phase_map, 3, AB
CODE, phase_map, 5, AC
CODE, phase_map, 6, BC
CODE, phase_map, 7, ABC
CODE, phase_map, A, A
CODE, phase_map, B, B
CODE, phase_map, C, C
CODE, phase_map, AB, AB
CODE, phase_map, BA, AB
CODE, phase_map, AC, AC
CODE, phase_map, CA, AC
CODE, phase_map, BC, BC
CODE, phase_map, CB, BC
CODE, phase_map, ABC, ABC
CODE, phase_map, CBA, ABC
CODE, phase_map, BCA, ABC
CODE, phase_map, BAC, ABC
CODE, phase_map, CAB, ABC
CODE, phase_map, Unknown, ABC
CODE, phase_map, Null, ABC
DEFAULT_CODE, phase_map, ABC
```

This example is for using a lookup table (codelist) that is stored in a database table.

```
# CODE feeder_ncg
#
DBCODE, feeder_ncg, feeder_name, ncg_id
DEFAULT_CODE, feeder_ncg, --INTEGER_SEQUENTIAL--
```

This example is for using a single lookup table (codelist) that is stored in a database table where you need multiple fields returned.

```
# CODE pf_capacitor_data_kvar_rating_a
#
NAMED_DBCODE, pf_capacitor_data_kvar_rating_a, pf_capacitor_data,
catalog_id, kvar_rating_a
DEFAULT_CODE, pf_capacitor_data_kvar_rating_a, 0

#
# CODE pf_capacitor_data_kvar_rating_b
#
NAMED_DBCODE, pf_capacitor_data_kvar_rating_b, pf_capacitor_data,
catalog_id, kvar_rating_b
DEFAULT_CODE, pf_capacitor_data_kvar_rating_b, 0

#
# CODE pf_capacitor_data_kvar_rating_c
#
NAMED_DBCODE, pf_capacitor_data_kvar_rating_c, pf_capacitor_data,
catalog_id, kvar_rating_c
DEFAULT_CODE, pf_capacitor_data_kvar_rating_c, 0
```

Model Build Workbooks

The core model preprocessor configuration files are maintained and generated from the two workbooks, the Oracle NMS Distribution Modeling workbook and the Oracle DMS Power Flow Engineering Data workbook.

Distribution Modeling Workbook

The modeling workbook contains many tabs to map a customer's GIS data to the standard NMS model. These tabs include device-mapping tabs, attribute-mapping tabs, and symbology mapping tabs. Mapping is accomplished by assigning each GIS object an NMS class based on specified criteria. Attributes associated with the GIS objects mapped are then also mapped to NMS attributes in their appropriate attributes tab. The mapping information entered into these tabs will be used by the NMS Model Config Generator to generate a set of customer specific model and preprocessor configuration files; this workbook is used as a container to store configuration.

The System Distribution Model workbook maintains configuration for the following files:

- Classes File
- Inheritance File
- Attribute Schema File
- Attribute Configuration File
- State Mapping File
- Voltage Symbology File
- Rank Configuration File
- Hide/Display File
- Declutter File
- Electrical Layer Objects File
- Landbase Layer Objects File

Model Configuration Files Generated by the NMS Model Config Generator

The NMS Model Config Generator is a stand alone application that is used to read the Distribution Modeling workbook, parse the contents, and generate the desired configuration files. Below is a list of all files that can be generated with a brief description. Notice that [project] indicates that the files generated pertain to a specific project configuration.

File	Description
[project]_classes.dat	Contains all NMS classes being used in the current workbook mapping.
[project]_inheritance.dat	Contains the inheritance structure of all classes being used in the current workbook mapping. This structure may include NMS required inheritance definitions.
[project]_schema_attributes.sql	Contains the schema definition for all attributes in the NMS Model. Along with the schema definition, a view is also defined for each database table created. The view is created based on the display names provided in the attribute tabs.

File	Description
[project]_attributes.sql	Contains the attribute mapping specified in each of the attribute tabs. This mapping is used during model build time to insert the specified attribute mapping into the appropriate NMS model tables.
[project]_ssm.sql	Contains a symbol to device mapping based on the nominal and current states of the device.
[project]_devices.cel	Contains the actual mapping criteria definition for all electrical devices. The criteria are derived from the information in the mapping tabs.
[project]_landbase.cel	Contains the actual mapping criteria definition for all landbase objects.
[project]_declutter.sql	Contains the configuration that defines the level at which objects disappear and reappear in the Viewer.
[project]_pf_symbology.sql	Contains the coloring information for each of the PF specific views available in the Viewer (*e.g.*, View Currents, View Loading, etc).
[project]_schema_attributes_view.sql	Contains the SQL view configuration for the different attribute tables. This configuration is used by the NMS Attribute Viewer.

Mapping Tabs

There are ten object-mapping tabs in the workbook. These tabs are used to specify the GIS object and the exact criteria for a GIS object to map to the selected NMS class. Below is a list of all the mapping tabs with a brief description.

Workbook Tab	Description
Core Nodes	This tab contains all NMS core nodes. These core nodes are used during CELL file generation. They will not be included in the classes and inheritance files.
Devices	Intended for the mapping definition/criteria of all electrical objects (Switches, Transformers and other operable devices).
Conductors	Intended for the mapping definition/criteria of all conductor objects.
Customer & Service	Intended for the mapping definition/criteria of all electrical service devices. Such as point of service, generators and meters.
Structures	Intended for the mapping of structure objects, such as manholes, poles and switchgear cabinets.
Landbase	Intended for mapping of all background parcel data.
Annotation	Used to map text objects from both the electrical and background layers to specific SPL classes.

Mapping Syntax

To take advantage of the tools included in the workbook, the correct syntax must be used. The workbook is to be mapped using a simpler syntax than the CELL explosion language. When in doubt about specific syntax, you can always assume that if it conforms to the CELL explosion language, it will work for the workbook mapping.

Class Mapping Columns and Syntax

Column	Description
Parent Class	This is a locked column and should only be modified by NMS model engineers. This column is used to define the inheritance lattice. The class in this column defines the parent for the child found in the next column "Class Name". Multiple parents can be defined for a single class using a comma "," to separate the class names.
Class Name	This is a locked column and should only be modified by NMS model engineers. This column indicates the name of the class.
Attribute Table	This is a locked column and should only be modified by NMS model engineers. This column indicates the table in which the attributes associated with this class will be stored.
Class Number	This is a locked column and should only be modified by NMS model engineers. The number in this column indicates the class number of the NMS class.
Index	This column is used to specify the index to be used during CELL file generation. The syntax for this column is CELL explosion language syntax. The CELL file generated will always use attribute index (ATTR_INDEX) to specify an index for a specific object using the data found in this column. Example: [ATT_TransformerOH.OBJECTID]
Phase	The criteria specified in this column will be used during CELL file generation to specify a phase value to the device being processed. If this column is left blank, ABC phase will be used. Example: [ATT_TransformerOH.PHASES]
Nominal Status	The criteria specified in this column will be used during CELL file generation to specify the nominal status of the device as it is being processed. If this column is left blank, CLOSED will be used. Example: [ATT_TransformerOH.NORMALLSTATE]
NCG	The criteria in this column will be used during CELL file generation to indicate the network control group of the device being processed. Example: [%feeder_ncg.[ATT_TransformerOH.[CIRCUITID]]
From_Node	The criteria in this column will be used during CELL file generation to indicate the topological from connection. Example: [OBJ_PORT_A]
To_Node	The criteria in this column will be used during CELL file generation to indicate the topological to connection. Example: [OBJ_PORT_B]
Physical Properties	The criteria in this column will be used during CELL file generation to specify the special characteristics of this device such as lateral or backbone. Example: [%phys_prop.[ATT_TransformerOH.PROPERTIES]]

Column	Description
Rank	The criteria in this column will be used during CELL file generation to specify the rank to be used for hide display configuration. Example: [%rank_bit_mask.[OBJ_CLASS]]
Capable Phases	The value in this pull down menu will be used during state mapping generation. It is used to indicate the possible phases a device can have. This information is important when generating the permutations needed for symbol mapping.
Gang Operated	The value in this pull down menu will be used during the generation of the inheritance lattice. If gang operated is selected, the class it is set for will contain an additional parent of "gang_operated".
Outage Stop Class	This value is not currently being used.
Symbology Enumerator	The criteria in this column will be used during CELL file generation to specify the symbology ID for the device. Note that this field is limited to 5 digits. By convention, we use CCCEP for electrical objects, where CCC= class number, E=enumerator (1-9), and P= phase. Example: 1050[%phs_num.[ATT_TransfomerOH.PHASES]]
Coordinate Definition	The criteria in this column will be used during CELL file generation. The CELL file generated will always use relative coordinates. If absolute coordinates are require, then the ABSOLUTE_COORDS, 1 key word must be specified. If this column is not populated then the following will be used: COORD1, 0, 0 COORD2, 0, 10 Example: ABSOLUTE_COORDS, 1 COORD1, [ATT_X1], [ATT_Y2] COORD2, [ATT_X2], [ATT_Y2]
Add Text Mapping	The values in this column should only be added through the text-mapping window. The window starts by clicking on the column button ("Add Text Mapping"). Specify the row and column for the class the mapping is intended for. All information in the form is to be entered using CELL file syntax. The information entered for the text class mapped will be saved to the tab "Text Mapping". Multiple text classes can be added for each class. When a text class is mapped and saved from the text-mapping window, the text class used will be populated in the "Add Text Mapping" column.
Alias Definition	The criteria in this column will be used during CELL file generation. Example: SW-[%sw_type.[ATT_Switch.FACILITY_TYPE]]
Display Name	The value in this column must be unique to the workbook and must not contain any spaces. This value is used as the display name for the control tool title.
GIS Object	The criteria in this column indicate the GIS object or feature class that will be used during the mapping in the CELL file (Example: MATT0, [ATT_TYPE], SWITCH). Multiple objects or GIS features can be separated by the "|" (OR) identifier. Example: SubstationDevices|CircuitBreaker

Column	Description
GIS Attribute that qualifies extraction	The criteria in this column indicate the GIS attribute to test on during the mapping stage. Multiple attributes can be used. Multiple attributes will be "AND" ed together. To indicate that multiple attributes are to be tested, a new line must separate the attributes. The OR condition cannot be used. Example: (AND) SubstationDevices.SUBTYPE SubstationDevices.SCADACONTROLLED
GIS Attribute criteria for extraction	The criteria in this column indicate the GIS attribute value that must be found for the expression to be true. Multiple values can be listed in an OR condition separated by the "|" character. For an AND condition, the values must be separated using a new line. The amount of new lines must match the number of new lines in the previous column. Example: CircuitBreaker SCADA Controlled
Comments	This column is intended for any additional comments desired to better inform the customer or model engineer of what is desired.
MP File Object	This column is not required. It is intended to provide more information about the object definition as found in the MP file.
MP Qualifying Attributes	This column is not required. It is intended to provide more information about the attribute names as found in the MP file.
Special Processing	This column is used to indicate that special processing exists for a particular device mapping. The "Special Processing" tab should be populated with the special CELL file criteria to be added to the mapping. The "Display Name" column is used to indicate the link to the "Special Processing" tab.
Comments	This column is intended for any additional comments desired to better inform the customer or model engineer of what is desired.

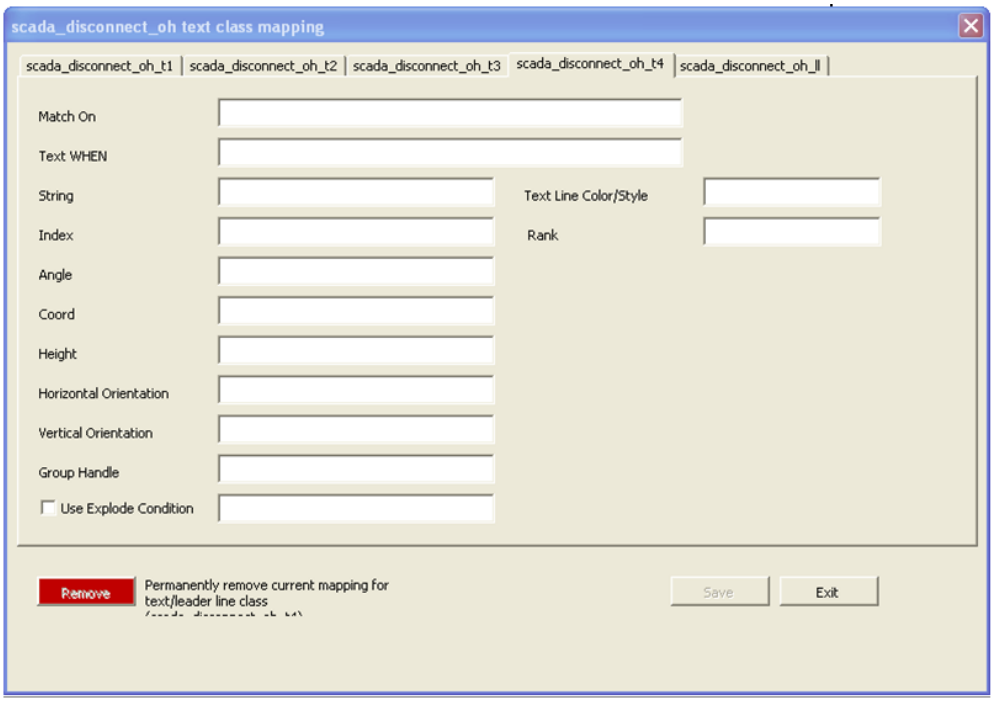
Note: The class will not be exported unless both the **Index** and **GIS Object** fields are populated.

Attribute Mapping Columns and Syntax

Column	Description
Attribute	The NMS model attributes being mapped. This column is locked and should not be modified.
Example Value	Example information, where appropriated. This column is locked.
Data Type	The data type of the attribute being mapped. This column is locked and should only be changed by an NMS model engineer.
Required / Recommended	Indicates if this attribute is required or recommended and indicates by which module the attribute is required or recommended. The color is used to indicate if it is required or recommended.
Field Order	If the model configuration is set to generate the attribute view definitions with supporting tables (see the **Tools** tab in the workbook), this field will be used to configure if the attribute should be included. If set to **NO_VIEW**, the attribute will not be included in the associated attribute view definition; if blank, it will be included.
Display Name	Specifies the name of the attribute, as it will be displayed in the Attribute Viewer. If one display name is set, it assumes all attributes will have a display name and uses the NMS attribute name if no display name is specified. Only attributes containing values will be displayed in the Attribute Viewer Tool.
GIS Class	Indicates the name of the GIS object or feature.
GIS Attribute	Indicates the name of the GIS attribute. This column is critical to correct attribute mapping in the CELL file. The prefix of ATT_ is not required for script style preprocessor as long as the "Use ATT_ Prefix" is selected in the "Tools" tab. Complex mapping should be done using lookups and/or conditional statements in CELL file syntax.
Comment	Used to specify additional information that may be useful to the modeler or customer.
MP File Objects	This column is not required. It is intended to provide additional information about the object as found in the MP file.
MP Qualifying Attributes	This column is not required. It is intended to provide additional information about the attribute as found in the MP file.
Special Processing	This column is not required.
Comment	Used to specify additional information that may be useful to the modeler or customer.

Text Mapping Window

The text-mapping window is to be used for text mapping when the text to be displayed is not included in the data as a separate object. This is true for most attribute based annotation GIS systems. The screen capture below is an example of how a text object can be created for a device class based on the value of an attribute.

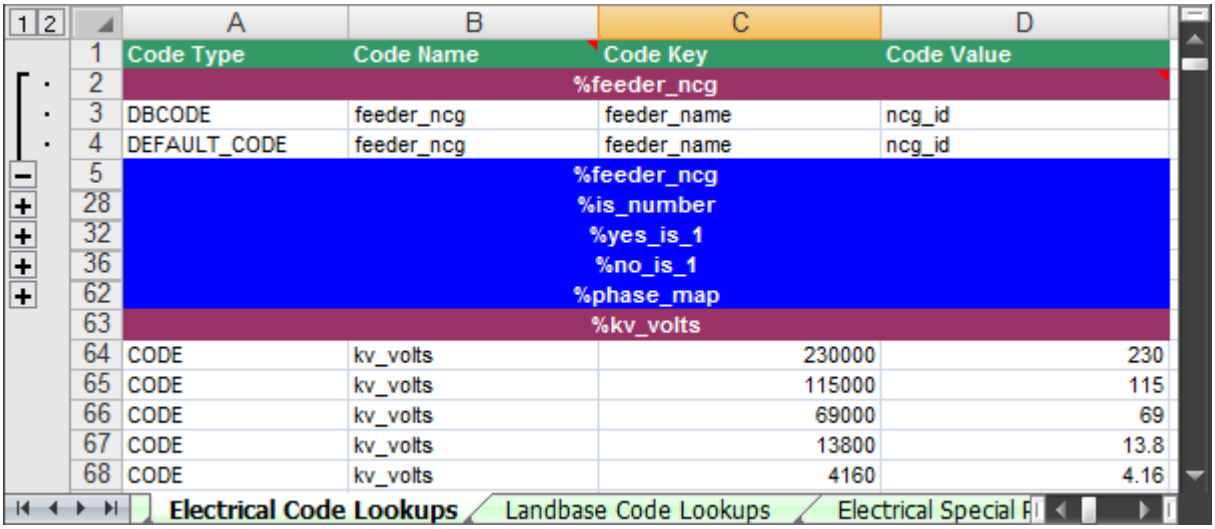


Code Lookups

All code lookups to be used in the mapping of the workbook must be specified in their appropriate tab in the workbook. This information is to be entered by the NMS model engineer. Lookups can be database lookups by specifying them as db code lookups in the appropriate CELL file syntax.

Electrical Code Lookups	Contains lookups to be included in the Electrical Layer Objects Cell File.
Landbase Code Lookups	Contains lookups to be included in the Landbase Layer Objects Cell File.
Gas Code Lookups	Contains lookups to be included in the Gas Layer Objects Cell File.

Code Lookups Example



	A	B	C	D
1	Code Type	Code Name	Code Key	Code Value
2	%feeder_ncg			
3	DBCODE	feeder_ncg	feeder_name	ncg_id
4	DEFAULT_CODE	feeder_ncg	feeder_name	ncg_id
5	%feeder_ncg			
28	%is_number			
32	%yes_is_1			
36	%no_is_1			
62	%phase_map			
63	%kv_volts			
64	CODE	kv_volts	230000	230
65	CODE	kv_volts	115000	115
66	CODE	kv_volts	69000	69
67	CODE	kv_volts	13800	13.8
68	CODE	kv_volts	4160	4.16

Electrical Code Lookups | Landbase Code Lookups | Electrical Special F

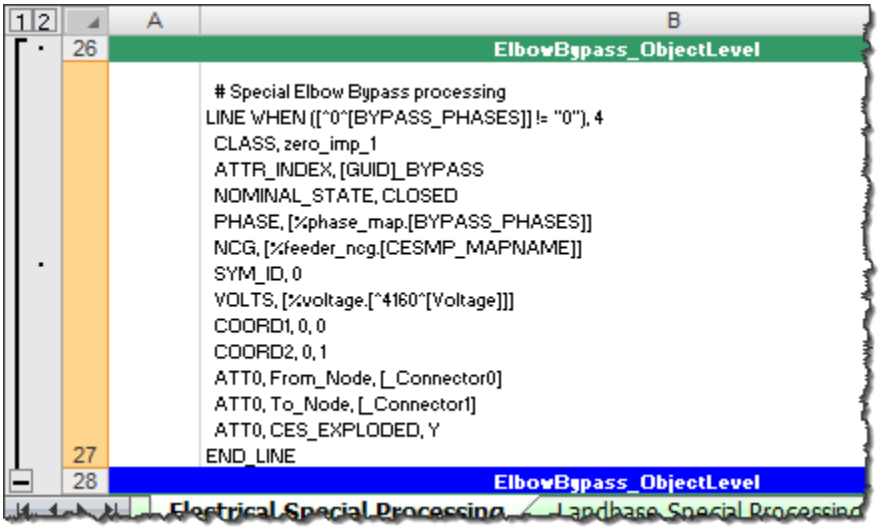
Special Processing Tabs

The Special Processing tabs are arranged according to the cell file they should be included in. A model engineer can use these tabs to add any special CELL file enhancement that cannot be fully generated by the workbook. This includes the addition of nodes such as FBD, FID, SRC, and SND nodes. There are two hooks for each CELL file block generated. One is at the device level, before the end of the first object's END_LINE or END_POINT). The second is before the cellblock is over, before the END_CELL.

To specify that special processing is required, populate the Special Processing tab in the appropriate class-mapping tab with the display name of the class that requires special processing. The special processing to be used must be specified in a single cell at the appropriate level in the appropriate tab. The level at which this is added is indicated by the name of the special processing section. An example is provided below:

Electrical Special Processing	Special processing for all electrical objects found in sheets, "Devices", "Customer & Service", "Structures", "Annotation" and "Conductors".
Landbase Code Lookups	Special processing for all land base classes found in sheet "Landbase".
Gas Special Processing	Special processing for all gas mapping sheets.

Special Processing Example



Adding an att_[*] Tab to the Model Workbook

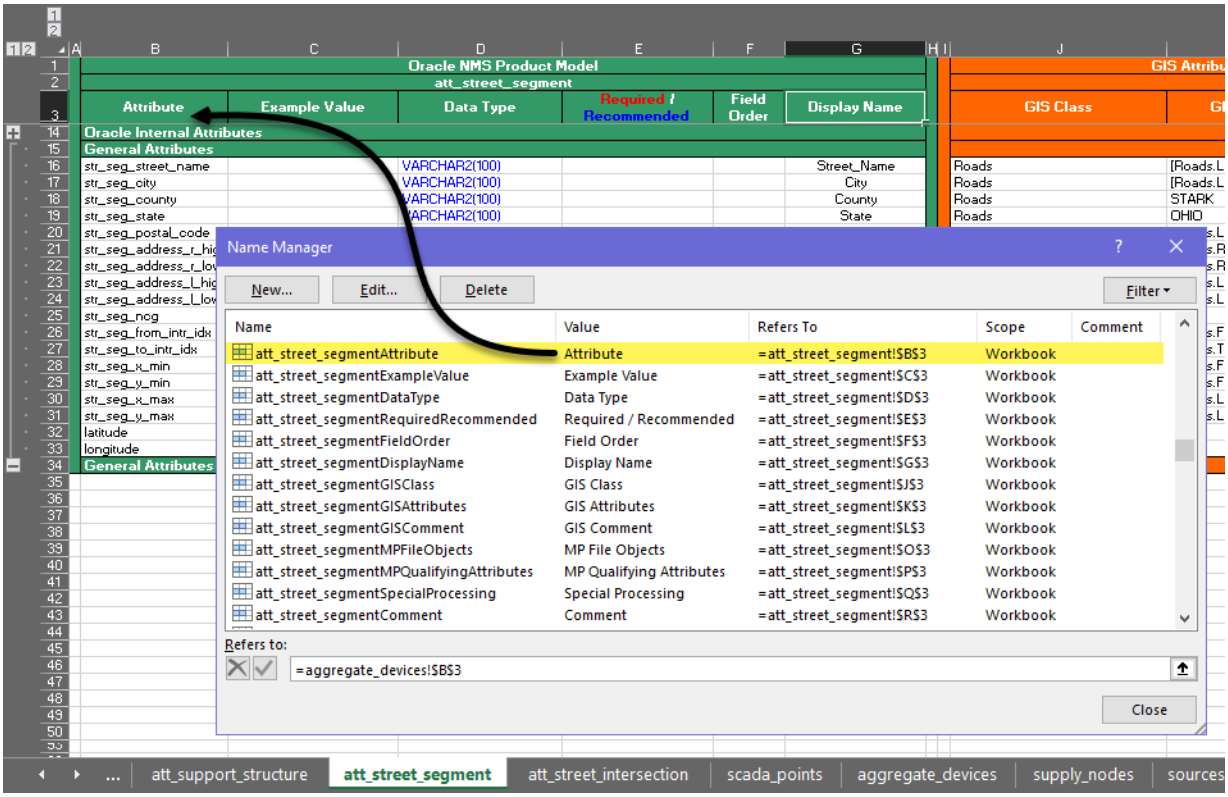
When a project specific device type is not already defined in the model workbook, you need to add an **att_[*]** tab to the workbook. To add the **att_[*]** tab, complete the following steps.

1. Copy the **attribute_template** tab.
2. Rename the copied tab using this convention: `att_[device/component to be modeled]`. For example, street
3. Defined names will need to be placed in the new tab so the Model Config Generator Tool can identify the valid column positions.
4. The columns on the third row of the tab need to contain the defined names.

   Note: Reference the other **att** tabs for examples.
5. From the Excel **Formulas** ribbon, click **Name Manager**.

   The Name Manager allows you to see and edit all defined names within the workbook.
6. Use the Name Manager to add defined names to each column in the new tab. The naming convention is the tab name concatenated with column name.

   Note: Please use other att tabs as template when creating the new defined names. For example, in the screen capture below, the name of the tab is **att_street_segment** and the first defined name is a concatenation of the tab name and the value in cell B3 (`Attribute`), which results in a name of `att_street_segmentAttribute`.



7. Once the defined names have been created, configure the **att_** tab as needed.
8. Use the Model Config Generator tool to generate the newly created configuration.

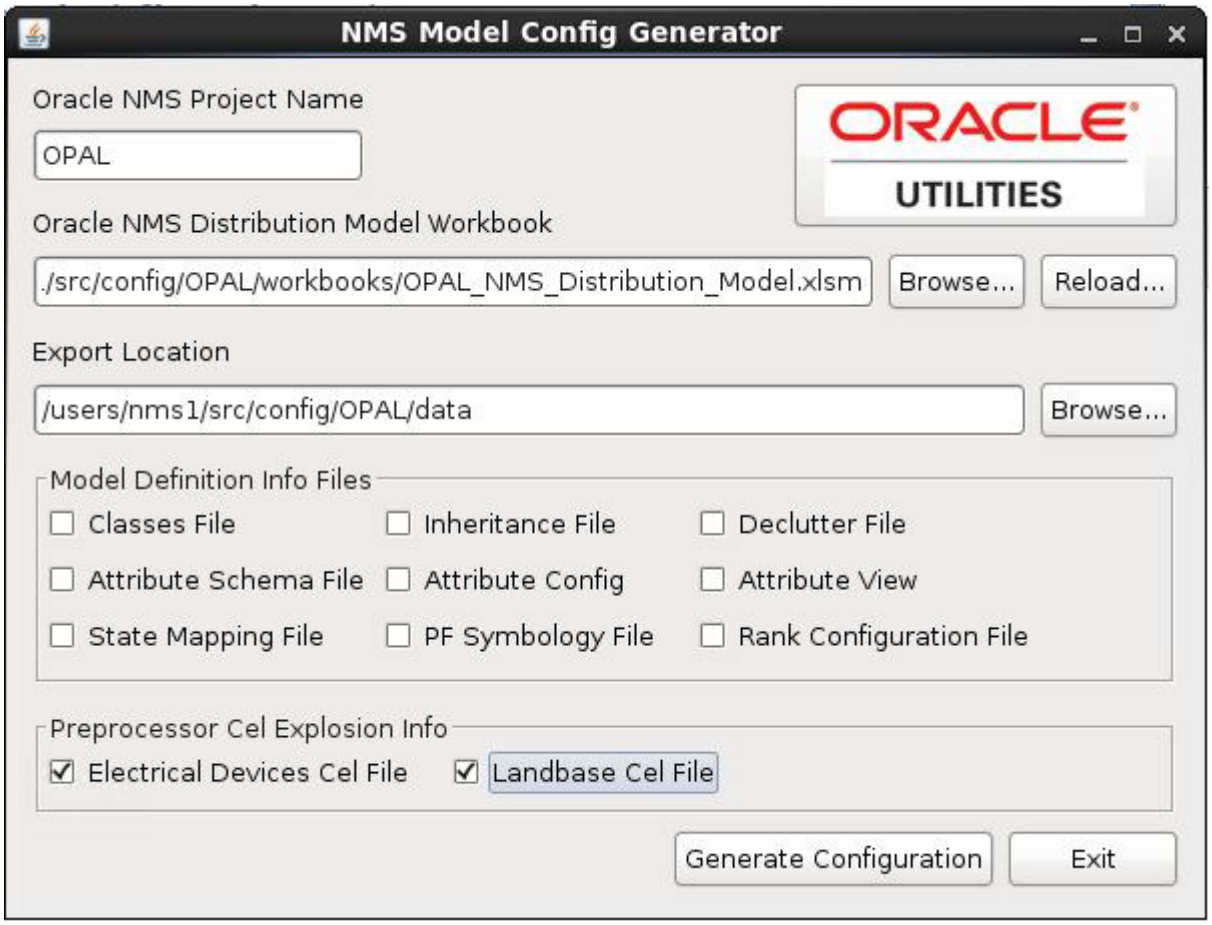
DMS Data Scripts

Project specific configuration needs to be set up during implementation to define which data files generated from the Power Flow Engineering Data Workbook should be loaded to the database during an NMS setup. This is done using a Korn Shell script that lists the DMS data SQL files that need to be loaded to the database and any ancillary configuration that may be needed by a project. The script needs to be project specific since device types present in the field will vary from customer to customer. You can use `OPAL-pf-setup` as a template for getting started; copy it to `[project]-pf-setup`. Lines can be added or removed, as needed.

A project specific version of `[project]-pf-views.sql` also needs to be created to specify configuration related to device limits and load profiles. There isn't a product version of this file since it will be unknown how load profiles will be configured or if temperature based limits will be used by a project; you can, however, use `OPAL_schema_pf_views.sql` as a template for getting started. Copy and rename it `[project]_schema_pf_views.sql` and then modify it to meet the project requirements. The main parts that need to be set up are related to the SQL view `PF_DEVICE_LIMITS`; if temperature based limits are to be used, this view will need to correspond to the number of temperature bands configured. If seasonal views are being used, the configuration will need to be based on the number of seasons configured.

Using the NMS Model Config Generator Tool

The Model Config Generator tool is used to generate model configuration files from a populated Distribution Modeling workbook. The tool is a standalone application that is responsible for reading the modeling workbook, parsing the contents, and generating the desired configuration files.



Generating Model Config Files

1. Make the appropriate changes to the Distribution Modeling Workbook.
2. After the changes are made, open the Model Config Generator Tool by using one of the following methods:
    a. On Unix, launch the generator using the following script:

    ```
    $NMS_BASE/bin/model-config-generator
    ```

    b. To run on a Microsoft Windows PC, your first need to zip the `NMS_BASE/bin/ModelConfigGenerator` directory by running the following command on a Unix terminal:

    ```
    zip -ry ModelConfigGenerator.zip $NMS_BASE/bin/
    ModelConfigGenerator
    ```

    Then move the zip file to the PC, unzip the file, and then navigate to the ModelConfigGenerator directory, and double-click the ModelConfigGenerator.jar file to launch it (assuming Java is installed).
3. Enter the project name, if it is not populated.
4. Browse to the location of the modified Distribution Model Workbook. The tool will read the workbook into memory. Once fully loaded, the **Export Location Browse...** button will become sensitized.

    Note: If a workbook has previously been opened and the path can still be reached, the tool will automatically select this path and load the workbook to expedite the process.
5. Browse to a location where you want to save the configuration files. Once the path is selected, the check boxes for the configuration files will become sensitized.

    Note: If an export path has previously been set, and the path can still be reached, the tool will automatically select this location to expedite the process.
6. Select the files you wish to generate.
7. Click the **Generate Configuration** button. The Model Config Generator Tool will generate the desired configuration files and create reports for any errors encountered.
8. If updates are made to the workbook, click the **Reload...** button to force a reload of the newly populated data.

Model Build Process for Work Orders

When running the model build process, there can be multiple versions of a given map in the queue of maps to be processed. Map versions must be processed in the order they are submitted to the model build process. If an older version of a map cannot be committed to the model, the system must keep the newer version from being applied.

For single map (version) processing, this is generally not an issue since all the files to be preprocessed are put into one directory; however, when the maps to be processed are provided in model build directories (work orders), the maps that cannot be committed (blocked work orders) will become dependencies on any future map or work order that contains any map from the blocked work order.

Configuration of this feature is optional and will be enabled if you define MP_DIRECTORIES and MP_EXTENSIONS. MP_DIRECTORIES will be a list of directories where your model import files are located (`export MP_DIRECTORIES=$NMS_HOME/data/mp`). MP_EXTENSIONS will be a list of extensions for import files (`export MP_EXTENSIONS=mp`). If you have multiple extensions or directories, delimit them with either semicolons (;) or spaces.

To configure the preprocessor to support work_order directories, follow the example in `OPAL-build-map` and make special note of the use of the environment variable `_wo_dir`, which is set by the `[project]-build-map` script to support work order style multi-partition directory builds. This variable will identify the `$OPERATIONS_MODELS/patches` sub-directory to write the .mb files to.

Power Flow Engineering Data Workbook

The Power Flow Engineering Data Workbook is an Excel spreadsheet used to gather and manage data required by DMS applications that are not generally available within the GIS and Oracle Utilities Network Management System. The Power Flow Engineering Data Workbook maintains data required to run the Power Flow, Suggested Switching, Optimal Power Flow, Feeder Load Management, Fault Location Analysis, and Fault Location Isolation and Service Restoration applications.

The Power Flow Engineering Data Workbook defines the required data types, the data tables, and the table schemas. An MS Excel spreadsheet is used for each data type and its corresponding data table. Tabs (worksheets) in the Excel spreadsheet contain a description of the data table and the data table columns. Each data worksheet also contains one or more user-editable tables the user fills for each device type in the data model. The user simply edits the enterable table, adding a new row for each unique device type. The engineering data entered into these tabs will be used by the Power Flow Data Generator to generate a set of customer specific engineering data configuration files.

The Power Flow Engineering Data workbook contains tabs for the following device types:

- Equivalent Sources
- Power Transformer Impedance 2 Winding
- Power Transformer Impedance 3 Winding
- Power Transformer Tap Data
- Power Transformer Limits
- Switch Fuse Limits
- Customer Loads
- Customer Hourly Profiles
- Capacitor Banks
- Distributed Energy Resources

- Line Phase Impedances
- Line Sequence Impedances
- Overhead Line Construction
- Overhead Line Conductors
- Overhead Line Framing

Data Files Generated by the Power Flow Data Generator

The Power Flow Data Generator is a stand alone application that is used to generate the desired data files. Below is a list of all files that can be generated with a brief description. Note that **[project]** indicates that the files generated pertain to a specific project configuration. The file generated should be installed in the respective runtime SQL directory as well as the project configuration SQL directory.

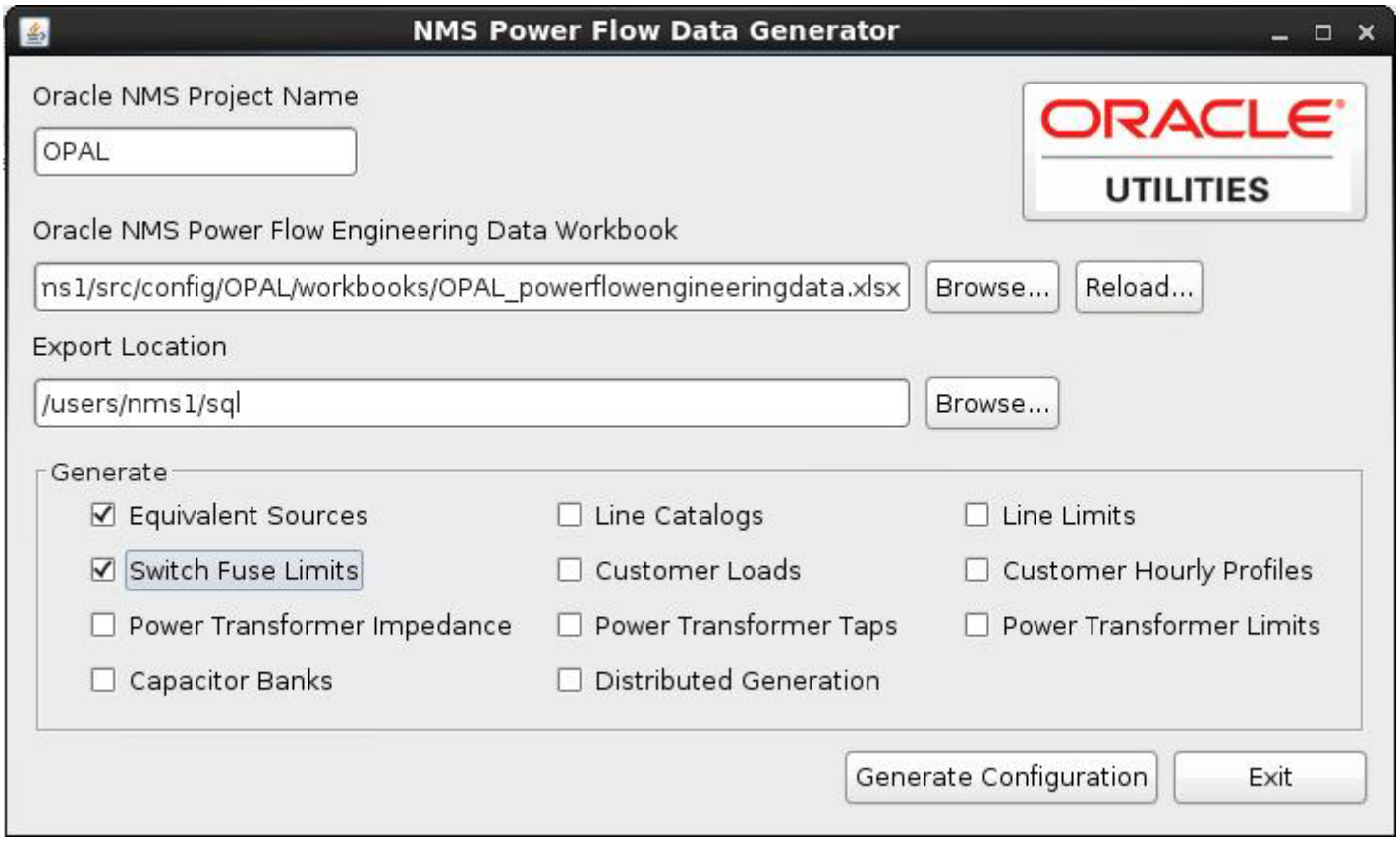
Please note that for new DMS customers, some product SQL files may not run into the database properly until the below data SQL files are added to the project configuration. When initially setting up a new DMS data model, the SQL files below can be generated with no data such that the table schema's can be run into the database during the initial nms-setup.

Data file	Description
~/sql/ [project]_powerflowengineeringdata.xlsm	This is the latest checked-in version of the Power Flow Engineering Data workbook, to be used for generating the customer catalog data sql files.
~/sql/[project]_pf_sources.sql	Contains data pertaining to equivalent source models for the source nodes in the network.
~/sql/[project]_pf_line_catalog.sql	Impedance details of lines.
~/sql/[project]_pf_line_limits.sql	Line limit details.
~/sql/[project]_pf_switches.sql	Contains nominal ampacity data for switches.
~/sql/[project]_pf_load_data.sql	Contains electrical characteristics of customer loads.
~/sql/[project]_pf_xfmrtypes.sql	Contains electrical characteristics data for power, step and auto transformers.
~/sql/[project]_pf_xfmrtaps.sql	Contains electrical characteristics data for power, step and auto transformers.
~/sql/[project]_pf_xfmrlimits.sql	Contains multiple ratings/limits for branch flows based on seasons for transformers.
~/sql/[project]_pf_capacitors.sql	Contains electrical characteristics of capacitors and reactors.
~/sql/[project]_pf_hourly_load_profiles.sql	Contains profiles for load and distributed generation classes. For load profiles this may consist of Residential, Commercial, Industrial. For dist gen if may consist of PV:Sunny, PV:Cloudy.

Data file	Description
~/sql/[project]_pf_dist_gen_data.sql	Contains electrical characteristics of distributed generation devices.

Using the NMS Power Flow Data Generator Tool

The Power Flow Data Generator tool is a tool used to generate engineering data files from a populated Power Flow Engineering Data workbook. The tool is a standalone application that is responsible for reading the workbook, parsing the contents, and generating the desired configuration files.



Generating Data Files

1. Make the appropriate changes to the Power Flow Engineering Data workbook.
2. After the changes are made, open the Power Flow Data Generator Tool by using one of the following methods:
    a. On Unix, launch the generator with the following script:

    ```
    $NMS_BASE/bin/powerflow-data-generator
    ```

    b. To run on a Microsoft Windows PC, your first need to zip the `NMS_BASE/bin/PowerFlowDataGenerator` directory by running the following command on a Unix terminal:

    ```
    zip -ry PowerFlowDataGenerator.zip $NMS_BASE/bin/
    PowerFlowDataGenerator
    ```

Then move the zip file to the PC, unzip the file, and then navigate to the `PowerFlowDataGenerator` directory, and double-click the PowerFlowDataGenerator.jar file to launch it (assuming Java is installed).

3. Enter the project name, if it is not populated.
4. Browse to the location of the modified Power Flow Engineering Data Workbook. The tool will read the workbook into memory. Once fully loaded, the **Export Location** field **Browse...** button will become sensitized.

   Note: If a workbook has previously been opened and the path can still be reached, the tool will automatically select this path and load the workbook to expedite the process.
5. Browse to a location where you want to save the configuration files. Once the path is selected, the configuration file check boxes will become sensitized.

   Note: If an export path has previously been set, and the path can still be reached, the tool will automatically select this location to expedite the process..
6. Select the files you wish to generate.
7. Click the **Generate Configuration** button to generate the configuration files. The Power Flow Data Generator Tool will generate the desired configuration files and create reports for any errors encountered.
8. If updates are made to the workbook, click the **Reload...** button to force a reload of the newly populated data.

Model and Power Engineering Workbook Locations

An example of these workbooks is included in the Oracle Utilities Network Management System Oracle Power and Light example model and configuration included with every release package. You can find these two workbooks in the `$NMS_BASE/OPAL/workbooks` directory of the Oracle Utilities Network Management System system.

Load Shed and Restoration

Feeder Load Groups Data Import

The Load Shed/Restoration application is driven by data in the LOAD_GROUPS and LOAD_GROUP_FEEDERS database tables. Those tables can be populated by any mechanism that is suitable for the business process and data sources being used by the project.

Care must be taken when updating the data in these tables because some columns (like last shed/restore times) are updated by the application. Therefore the import process must only update pertinent information in an existing entry. It is also advisable that the update be performed as part of the regular model build process and should be included in the post-build script.

A script (`nms-loadFeederLoadGroups`) can be used to populate the tables with data from a CSV file. This script expects the CSV files at the path `$OPERATIONS_MODELS/load_groups`. The processed CSV files are moved to `$OPERATIONS_MODELS/load_groups/parsed_CSVs` sub directory and the errors are logged at `$OPERATIONS_MODELS/load_groups/log` sub directory.

The CSV file format is specified in the LOAD_GROUPS_CONFIG database table. This includes configuring information like number of columns, column positions and whether header present or not

The script can be run with following command line options:

```
nms-loadFeederLoadGroups [-clean] [-v] [-debug]
```

-clean	This will delete all the data in load groups tables and load the groups from CSV file into these tables.
-v	Turn on verbose output for application.
-debug	This will turn on debug for assisting with issues in the adapter. If debug is turned on it will by default turn on verbose as well.

The following assumptions apply:

1. The input CSV files are stored in the `$OPERATIONS_MODELS/load_groups` directory.
2. Once they have been processed, these files are moved to the `$OPERATIONS_MODELS/load_groups/parsed_CSVs` directory.
3. For duplicate feeder items within a single input file, the first item read will be used. In some instances theses duplicates may be flagged as errors and placed into the error log file.
4. Log files will be placed in the `$OPERATIONS_MODELS/load_groups/logs` directory.
5. When adapter is executed any old log files will be moved to the `$NMS_HOME/data/load_groups/logs/old_log` directory.

Log Files Generated

`Load_Groups_Data_Errors_YYYYMMDD_HHMMSS.log`: This file will contain any invalid data rows along with an error code from the below list.

- LG-001 # Too many fields
- LG-002 # Too few fields
- LG-003 # Invalid initial priority
- LG-004 # Invalid device type
- LG-005 # Invalid load

To run the import script run during nms-setup, it should be included in [project]-pf-setup.

The OPAL configuration can be examined for examples of how this script is used. In OPAL, the scripts is called from a parent script (`LoadOPALFeederLoadGroups`).

The `LoadOPALFeederLoadGroups` script itself is called from `OPAL-pf-setup` and `OPAL-postbuild`.

Critical Customer Types

An SRS rule loadShedAffectedCustomerTypes is used to specify the critical customer types for Load Shed and Restoration. Critical customers that are impacted by Feeder Groups is displayed to the operator. The critical customer types included in the counts are specified by this rule. Typical value for rule_value_1 is "egkmt".

If all critical customers should be considered, then leave the rule_value_1 blank.

Model Manipulation Applications and Scripts

After a customer has built a model, there may be times when certain scripts or applications may need to be run to clean up errors that have been introduced into the model or to remove obsolete devices or maps. There are several scripts and applications that exist to do this model manipulation. These scripts and applications are described below.

DBCleanup

Most customers should run the DBCleanup application periodically. It examines the modeling database tables and looks for duplicate active rows, orphaned objects, and inconsistencies in the ALTERNATE_VIEWS table. If any of these problems are discovered, the application will attempt to fix the data so that it is consistent with the rest of the database tables.

Command Line Options

Command Line Option	Description
-debug / -d	Turn on debug / print out all SQL commands as executed.
-fixAll	Run all of the model cleanup functions, except for -fixICP.
-fixAVTable / -a	Fix alternate_views table. **Notes:** Check the ALTERNATE_VIEWS table for missing rows (in other words, the object is in DIAGRAM_OBJECTS multiple times but not in ALTERNATE_VIEWS. The number of rows in ALTERNATE_VIEWS for an object should be one less than the number of rows in DIAGRAM_OBJECTS.) or erroneous rows (object is in ALTERNATE_VIEWS multiple times with two different primary partitions). In the latter case the row will be deleted; in the former case, missing rows will be added.
-fixDevicesWithNoPartition / -p	Remove objects with no owning partition. **Notes:** Checks model tables (ALIAS_MAPPING, DIAGRAM_OBJECTS, ALTERNATIVE_VIEW, POINT_COORDINATES, core tables, and any attribute tables maintained by the model build) for bad partitions found in core tables (NETWORK_COMPONENTS, NETWORK_NODES and OBJECT_INSTANCES). When found, they will be removed.
-fixDuplicateRows / -r	Fix tables which have duplicate active rows or orphans. **Notes:** This option is the functional equivalent of the fixRedundant option plus the fixOrphans model. For efficiency, this process uses a slightly different algorithm.

Command Line Option	Description
-fixICP	Synchronize the pending construction & decommission conditions with the model.
-fixLocations	Check the integrity of the locations table and insert records as needed.
-fixOrphan / -o	Remove orphan aliases, attributes, diagrams. **Notes:** Checks some model tables (ALIAS_MAPPING, DIAGRAM_OBJECTS and POINT_COORDINATES, and any attribute tables maintained by the model build) for devices which are not currently defined in the model. When found, they will be removed.
-fixOrphanConditions	Remove electrical conditions which do not belong to an existing object.
-fixPartitions	Check the integrity of the partitions table and add missing records using the ALTERNATE_VIEWS table.
-help / -h / ? / -u	Print out a usage statement
-ignoreAttrTabs / -r	Do not attempt to repair attribute tables. **Notes:** Skips repair of the user attribute tables (ATT_FUSE, ATT_SWITCH, etc.) for both orphans and duplicates.
-listAffectedPartitions	Display the maps that were affected.
-mbdbs	Use MBDBService for database access.
-showMe / -s	Print out (don't execute) all SQL commands. **Notes:** All proposed transactions will be printed to standard out in SQL format. These may be saved to a file and executed.

See also Troubleshooting Issues with ICP Device Symbology on page 8-89 for fixing issues with ICP objects.

nms-delete-map

The `nms-delete-map` script allows the user to remove an obsolete map from the model. It creates a patch that is processed by MBService that will deactivate the map itself and all devices contained in it. This script should be used sparingly.

nms-delete-object

The nms-delete-object script allows the user to deactivate all instances of a single, specified device in all the maps in which it appears. It creates a patch that is processed by MBService to remove all the instances of the device.

nms-delete-patch

The nms-delete-patch script allows the user to delete a single patch or a range of patches that exist in the database. The script directly modifies the modeling database tables, potentially leaving the services in a state that is inconsistent with the current information in the database. After this script is used, either all services should be re-started or MBService should be re-started and then all the maps involved with the deleted patches should be re-built. After MBService re-builds the maps, it will send out notifications to the other services to bring them all into sync.

AuditLog

The AuditLog application works with the scripts and applications defined above to keep a persistent record in the database of the data manipulation activities that have been going on when a customer uses any of these scripts or applications. The information is stored in the `MODEL_AUDIT_LOG` database table and can be useful when trying to help support a customer with corrupted data by helping to provide a better scenario of the activities that might reproduce the problem.

Schematics

Oracle Utilities Network Management System– Schematics can automatically generate orthogonal schematic overviews of the nominal network.

Model Requirements for Schematics

In order to use Oracle Utilities Network Management System Schematics, the following is required of the data model:

- All substations must have the same partition class.
- The substation partition class must only be used by substations.
- All boundaries between feeders and substations are designated with a distinct class of devices.

Schematic Limitations

Since Oracle Utilities Network Management System Schematics uses a splayed-tree representation of the nominal network, it is necessarily geared towards radial networks and will have difficulty representing nominally looped, parallel or meshed areas. Oracle Utilities Network Management System Schematics is also geared towards simple network objects (for example, a switch) and cannot keep related devices in close proximity (for example, the internals of a switching cabinet).

Configuring Schematics

- All schematic configuration is controlled via command-line options which are passed to the schematic-generator, schematica. The script that contains the configuration is normally called [project]-create-schematics.
- The script must perform these three actions:
- Remove any previous schematic import files.
- Call schematica with all of the configured options.
- Process all generated import files.

The following table describes all of the command line options.

Command Line Option	Description
-addStop <list of classes>	Include these classes as well as those specified via -stop
-aliasType <alias types>	Change the priority for schematic annotation. This follows the same colon-separated convention as the PREFERRED_ALIAS environment variable. (Example: -aliasType GIS:OPS:PSU")
-allSubstations	The -allSubstations command will force the inclusion of all substations even if they do not have interconnected feeders or any feeders at all.
-balanceSubstations	Shift feeders around a substation until there are similar NUMBERS on each valid side.

Command Line Option	Description
-boundingBoxCls <class name>	Create a box of this class to indicate the substation-overviews extents. If unset, the box is not drawn.
-boundingBoxLabelCls <class name>	Create a label of this class, with the substation's name. Default is branch.
-branchWidth <dist>	Distance between two network branches that share a common upstream port. (see Figure 1 below)
-camelHumpHeight	Relative height (in terms of tier-height) of conductor-crossover bumps. Value between 1 and 0. (See Figure 4 below)
-camelHumpWidth	Relative width (in terms of branchWidth) of conductor-crossover bumps. Value between 1 and 0.
-deltasOnly	Only rebuild the specific maps that have changed since the last schematics build. Use this in your [project]-postbuild script if you feel that processing all schematics is too performance-intensive at the end of your model build.
-classesToLabel <list of classes>	List of classes for which the schematic-generator should create and place annotation.
-connectionClass <class name>	Device class to use when creating a branch to span two or more non-conductor devices. (Must be a non-electrical branch)
-coordSystem <#>	The coordinate system the schematic generator will assign all schematics. Should not be an existing value. Defaults to the current maximum coord_system + 1.
-dbname |-db <DBService prefix>	Force the schematic-generator to use the DBService that has the specified prefix (*i.e.*, -db MB will use MBDBService).
-dch	(Disable Camel-humps) Don't create camel-humps where conductors cross.
-defaultConductorSymbology <valid symbol class>	Use this symbol class when attempting to write out any conductor that has a symbol class of 0.
-defaultFeederDirection <north|south|east|west>	If the schematic-generator is unable to determine the direction for a feeder, it will use this value. No default. If this option is NOT specified, the schematic-generator will ignore any feeder for which it can not determine a direction.
-deviceGaps <class name> <scale factor>	Scales all diagrams of the specified classes by the specified amounts.
-deviceHeight <#>	Size of all non-conductor electrical branches. (See figures following this table.)

Command Line Option	Description
-deviceScaling <class name> <scale> <offset>	Scale all diagrams of the specified classes as well as shift them along their parent feeder's axis. Default scaling is 1.0, default offset is 0.0.
-excluded <class name>	Any classes specified here will be excluded from the generated schematic map with the exception of tie devices which will appear in the schematic view.
-fastCrossovers	Use a faster, but less accurate algorithm to determine where conductors intersect.
-fbdBounded	Use this option if all feeder-heads have FID on one port and an FBD on the other.
-feederDirection <north|south|east|west>	Have ALL feeders extend in the specified direction.
-feederHeight <#>	Minimum distance between a substation and the first device of a feeder. (See figures following this table.)
-feederNameTable <table name> <column name>	The specified table for each feeder's FID, annotated with the value found in the specified column. For single-circuit schematics, the feeder name is used as part of the generated map's name.
-feederOffset <#>	Distance between adjacent feeders. Default is branchWidth*10. (See figures following this table.)
-feederPrefix <comma-delimited list of strings>	Only process substations whose map names contain the specified strings.
-feederTextScale <#>	Amount to scale all feeder-name annotation.
-geographicSubstations | -gs <table name> <column name>	Use this option when all substations are modeled in the geographic world. Schematica will search the specified table for all classes listed in -substationNodeClasses and set the substation name based on the value in the specified column. **Note:** This option must be used in conjunction with -substationNodeClasses.
-globalScaleFactor <#>	Increases the size of all objects and all overviews by this amount.
-ignoreUnconnectedOpenPoints	Treat any open switch that has no connections on either port as if it were a regular switch when determining what devices to prune from the feeder schematic.
-intersubOffset <#>	Minimum distance between parallel sub-to-sub conductors. Defaults to tier-height or device-height*2, whichever is greater. (See figures following this table.)

Command Line Option	Description
-invisibleClasses <list of classes>	List of classes (that never have symbology, *i.e.*, zero-impedence conductors) that the schematic-generator should ignore when attempting to connect a feeder to its parent substation.
-labelClass <class name>	Use this text class when creating device annotation if the class <device_class_name>_t2 does not exist. Text class to use for all generated annotation. (See figures following this table.)
-mapPrefix <prefix>	Prepend all generated schematic maps with this prefix. **Required**.
-maps <list of map names>	The list of schematic map filenames (excluding file extension) that should be rebuilt, with spaces between the map names. Only these maps will be rebuilt.
-noFeederToFeeder	Do not connect up feeder-to-feeder tie-points. (See figures following this table.)
-noIntraFeederConnections	Do not connect up bypass tie-points.
-noPrune	Keep all devices in a feeder, not just those attached to open-points.
-noPruneCls <list of classes>	Keep all spurs that contain devices that belong to the list of classes. This option is unnecessary if running -noPrune.
-noPruneSCADA	Keep all spurs that contain SCADA devices. This option is unnecessary if already running -noPrune.
-noPrunePhysProperty <bitwise physical property integer>	Prevent devices with certain physical properties from being pruned. This is likely useful if you suspect that certain properties are not used consistently in the model and you want to be sure the schematic has, for example, all backbones, even if they are downstream from laterals. The integer argument is a bitwise OR of all the network_components.physical_properties bits that you want to prevent from being pruned.
-noSubstations	Do not draw substations. Instead draw all feeders in the same map in one or more rows. This option is only used when drawing all feeders in one map. To draw each feeder in a separate map, do not use this option; use the -substationName with an argument to group by feeder_name. (See -maxRowWidth)
-noSubToSub	Do not connect up sub-to-sub tie-points.

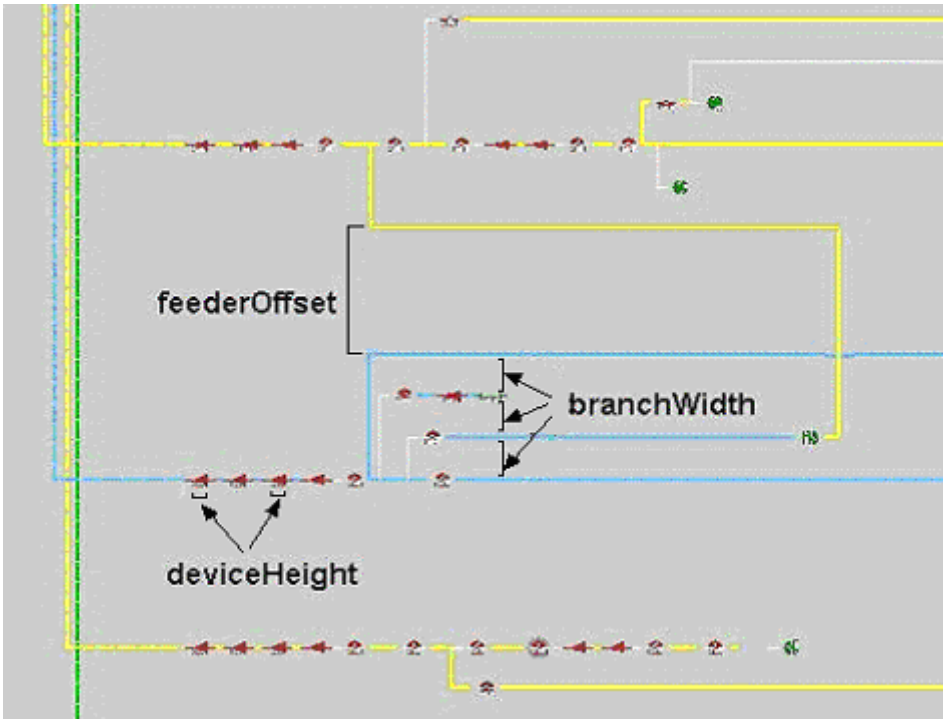
Command Line Option	Description
-orientation <ANY|HORIZONTAL|VERTICAL|ROUND_ROBIN|NONE>	Align all feeders according to the value. (ANY = normal feeder directions, HORIZONTAL = move all north/south-ward feeders to east/west, VERTICAL = move all east/west-ward feeders to north/south sides, ROUND_ROBIN = evenly distribute the feeders around the substation, NONE = move ALL feeders to side specified by "-feederDirection") Default is ANY.
-overlapConductors	Extend conductors to the center of adjacent switches. Default: false.
-overviewName <string>	The names of all resulting schematic maps will take the form: `<map prefix>_<overview name>_<substation name>`.
-placeSubsByConnection	Attempt to position substations with the greatest number of common connections closest to each other.
-priorityClasses <list of classes>	Keep the specified list of classes as close to the main trunk of the generated schematic tree as possible.
-ptns	A list of the partition numbers to query, rather than querying the entire model. This is used internally by MBService to process model edits quickly and should generally not be used in other cases.
-reorientDeviceClasses <list of classes>	Ensure that diagrams for the specified classes are always oriented from left to right. (Use this if symbols appear upside-down.)
-scaleFactor <#>	Multiplies the size of all conditions by this amount.
-singleCircuitStartClass	The single circuit schematic map can only start with one of these device classes. Choose these carefully and model circuits consistently. The resulting map name for this map will be the name of the FID that feeds this device.
-skipEmptyFeeders	Do not draw feeders that contain an exceedingly small number of devices (< 10 devices)
-sort <GEO|SPAN>	Arrange feeders either geographically (using only the anchor points of each feeder) or arrange them to minimize the distance feeder-to-feeder tiepoint connections must span. **Values:** GEO|SPAN • **GEO:** geographic ordering, • **SPAN:** minimal spanning tie points. Default is GEO
-startAtFID	Use when all feeder heads are modeled to have an FID attached.
-stop <list of classes>	List of all device or node classes the schematic-generator should not trace past.

Command Line Option	Description
-subSpacing <#>	Minimum distance between substations. No default. (See figures following this table.)
-substationBoxCls <class number>	Create a box of this class-type around the substation. No default. If not specified, there will be no visible box around the substation.(See figures following this table.)
-substationBoxSize <#>	Create a square of the specified size and scale the original substation schematic to place inside. Default is 1000. (See figures following this table.)
-substationName <database table name> <table column name>	Do not model substations. Instead, search the specified table for each feeder's FID and group them into substations based on the values in the specified column. For example, if the column is the feeder_name column, each feeder will have its own schematic map. It could also be the substation_name, which would group all feeders form one substation together. Or it could be any other column in the specified table or view, as desired.
-substationNodeClasses | -snc <list of class names>	When used in conjunction with -geographicSubstations, it specifies what type of nodes to initiate substation tracing from. Generally, the value should be SRC.
-substationPtnCls <#>	Only process substations with this specific partition-class. No default.
-substationTextScale <value>	Amount to scale the size of the substation label.
-substationTransitionClass <list of classes>	The set of classes that designate the transition between feeder and substation. Defaults to hyper_node.
-switchgeardevices <list of classes>	List of device classes for which schematic-generator should identify as switchgear devices.
-switchgearcolumnname <column name>	Column name of the base table for switchgear device classes, in which switchgear device name is populated. This switchgear device name gets displayed as annotated text over switch gear symbol.
-switchgearsymbolclass <class name>	Class number of switchgear symbol which gets displayed on switchgear devices.
-tapDeviceOffset <value>	Distance to offset single devices from the main trunk.
-textOffset <#>	Distance (along the feeder's main axis) to pull all device annotation. Default is 0. (See figures following this table.)
-textScale <#>	Scale all device annotation by this amount.

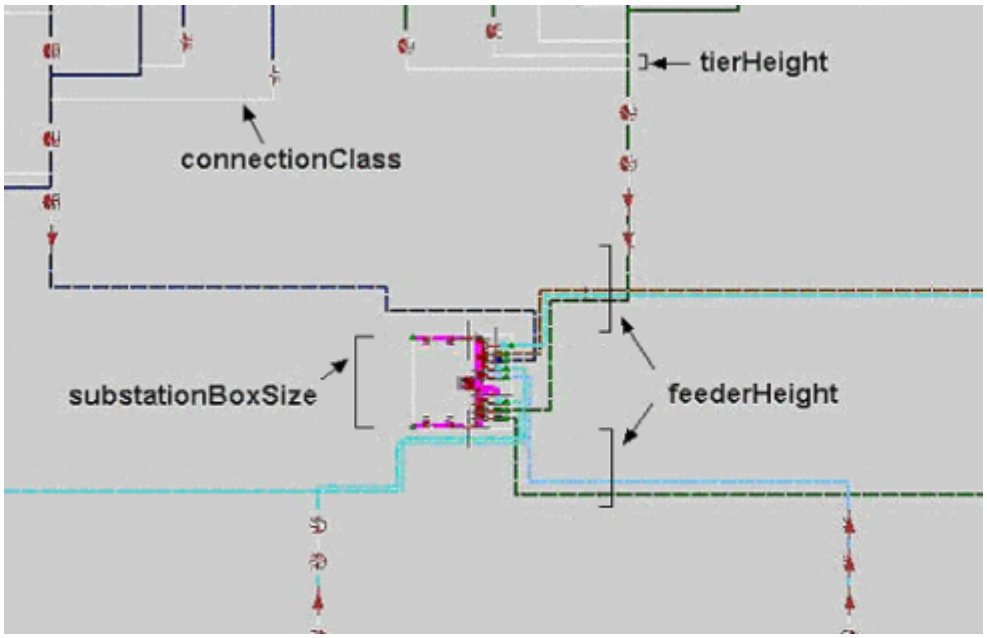
Command Line Option	Description
-textScaleSubstationDevices <value>	Amount to scale the text size for substation device annotations. Default value is 1.0. Alternative command is **-tssd**.
-tierHeight <#>	The distance (along the feeder's axis) a conductor will span. Default is 50. (See figures following this table.)
-tilebasedmaps | -tbm	Calculate the geographic orientation of each feeder based on the coordinates of the all open points, not on the base map's extents.
-validFeederStartClass <list of classes>	List of classes that designate the start of a feeder. **Required**.
-voltage <minimum voltage> <max voltage>	Only process devices that fall into the specified voltage range.
-weightClass [<class name> <weight>...>	Tells the schematic-generator to process certain classes of objects sooner when creating its internal schematic tree. If weight < 0, process later. If weight > 0, process sooner.

Note: <list of classes> format: [-]class name[+],[-]class name[+],....]
[-] exclude this class (and all descendents if '+' is used) [+] include all descendents.

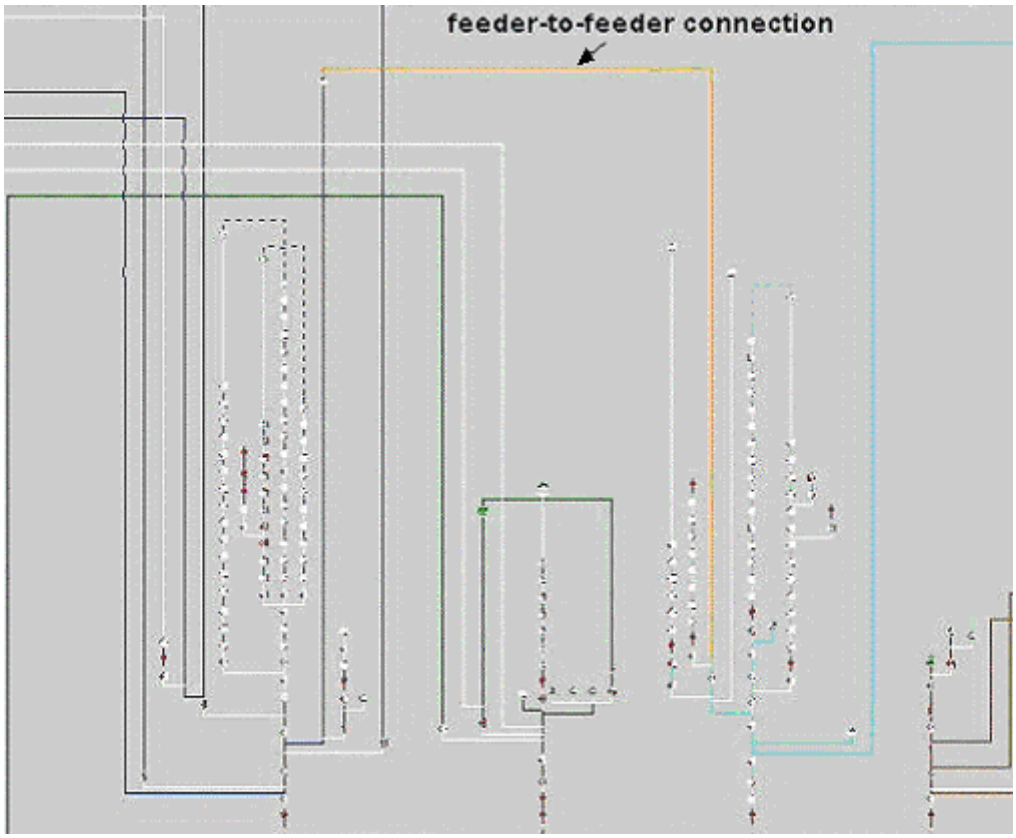
The following figure shows the deviceHeight, branchWidth, and feederOffset.



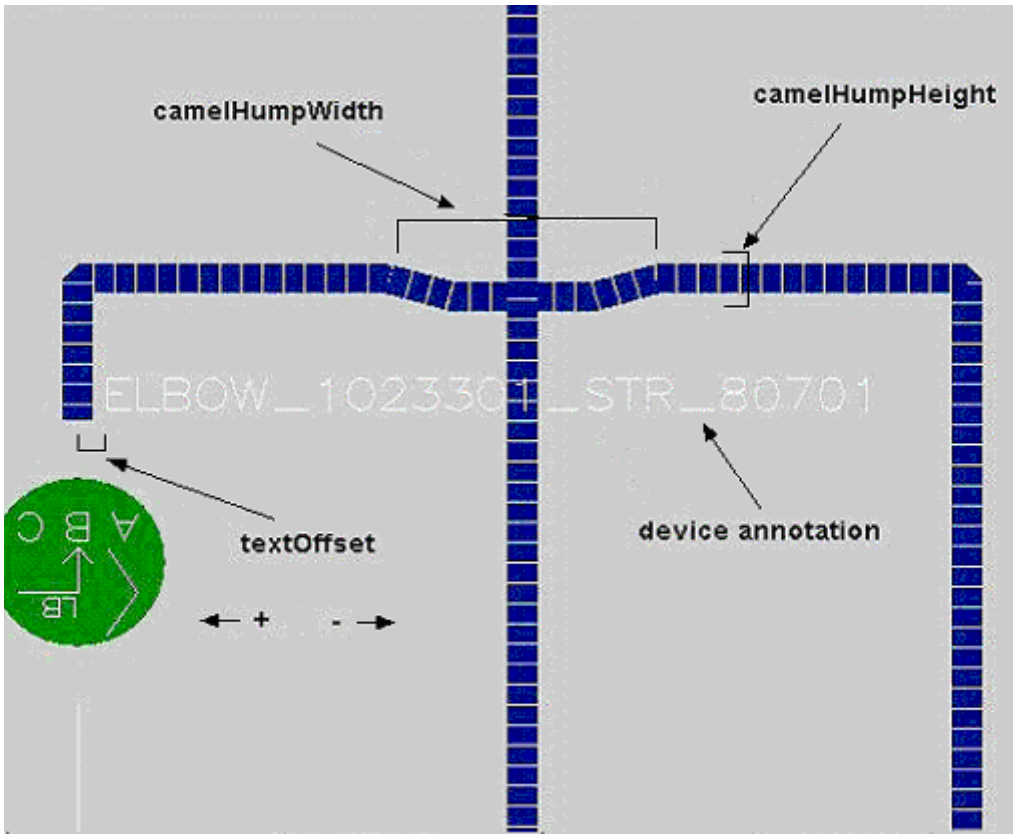
The following figure shows substationBoxSize, feederHeight, tierHeight, and connectionClass.



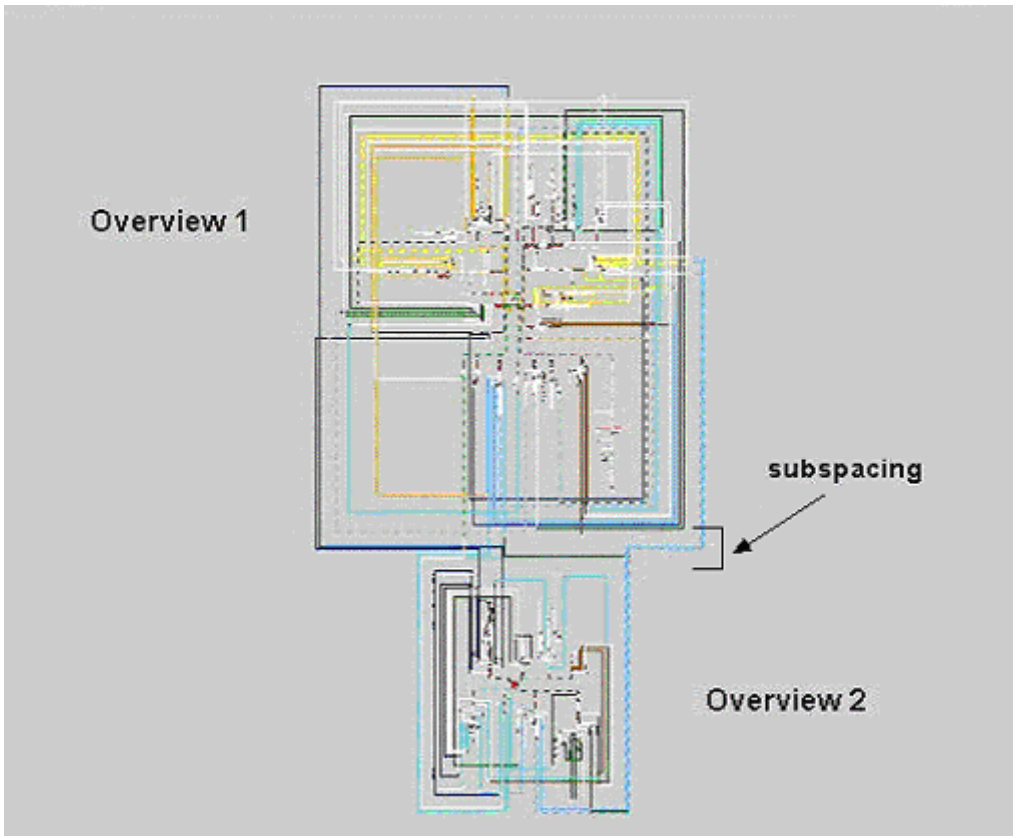
The following figure shows a feeder-to-feeder connection.



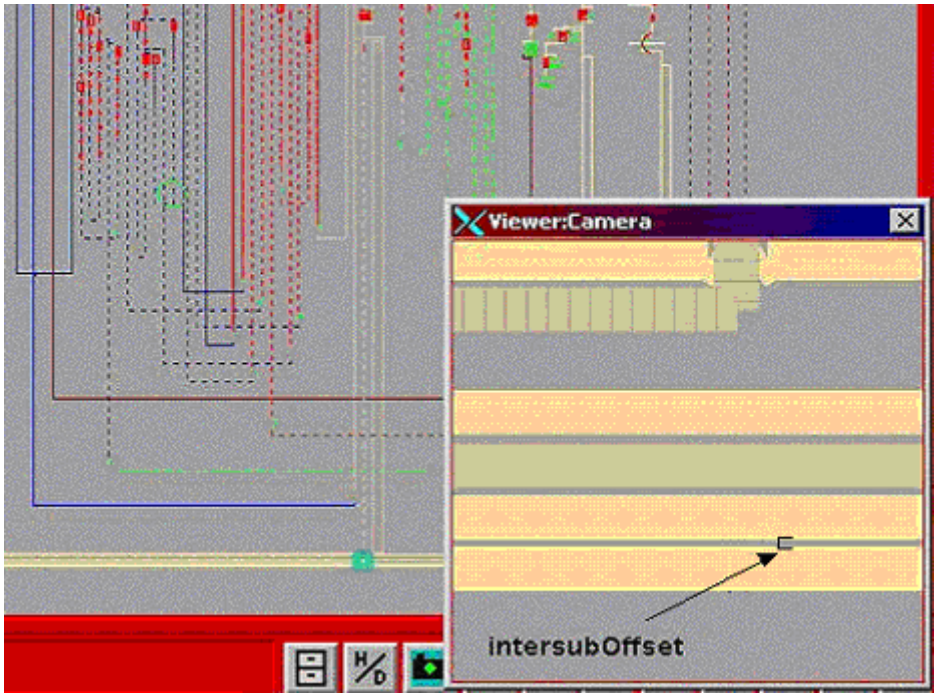
The following figure shows camelHumpWidth, camelHumpHeight, device annotation, and textOffset.



The following figure shows subspacing.



The following figure shows intersubOffset.



Schematic-Specific Symbology

Populate the new schematic_symb_class database table with any symbology state classes you wish to use in the auto-generated schematic maps. Use this if you want devices or conductors to appear different in the schematic maps than they do in the nightly build maps.

Note that this is non-product configuration; product does not add any schematic-specific symbology state classes to this table.

Generating Schematics

To create schematics, the customer-specific script [project]-postbuild must have a call to [project]-create-schematics.

The Post-Build Process

After the build process has processed the final map, it calls nms-postbuild. If there is a [project]-postbuild script, nms-postbuild will call it. If there is an entry for create-schematics, it calls it at this time.

Creating the Import Files

Once invoked, the schematic generator loads in the entire nominal network model and attempts to group all feeders with their parent substations. After it finishes determining the layout and spacing for all feeders and substations, it writes out one import file for each substation.

Processing the Import Files

After the schematic-generator creates the import files, the schematic script compares the most recent previous version of each file. If no changes are detected, it skips the map. Otherwise it proceeds to build the import file as per the normal model-build process.

Aggregate Devices

Aggregate devices are devices that are linked together in the model so that the user only needs to select one device and display one Control Tool to operate any number of associated devices.

Model Requirements for Aggregate Devices

Use the Distribution Model workbook to populate the aggregate_devices table. This can contain multiple records for a single controller device, so long as the seq_num is unique for each. In this manner, you can connect a single device to aggregate backfeed devices or replicate a more complex construct with a set of switches that all operate from a single Control Tool.

Field	Format	Comments
controller_cls	NUMBER(6)	The handle class of the controller id
controller_idx	NUMBER(12)	The handle index of the controller id
sec_cls	NUMBER(6)	The handle class of the secondary device
sec_idx	NUMBER(12)	The handle index of the secondary device
seq_num	NUMBER(6)	The sequence of the secondary device.

This is an attribute table, so the standard h_cls, h_idx, partition, birth, birth_patch, death. death_patch, and active columns are also required.

In Construction Pending / Device Decommissioning (ICP)

Oracle Utilities Network Management System supports the modeling and visualization of devices that are in-construction as well as devices that are marked for decommissioning. ICP can be used for commissioning new construction (such as road widening) and should not be used for nominal-state changes (such as feeder load balancing).

Device Lifecycles

In a GIS system, a device will fall in to one of four possible states:

Device State in GIS	Description
Install	Objects that are proposed construction or new objects to be commissioned at a future date
Existing	All objects that are in the GIS as-built and commissioned
Remove	Objects that are commissioned today and are part of the active model however there is a construction plan to remove these objects
Retired	All objects that have been completely de-commissioned. These devices will not exist in the real-time system.

Model Requirements for ICP

In order to use In Construction Pending (ICP), each affected device must have an additional value listed in their physical_properties entry inside the import file, as shown below:

Device State in GIS	Required Physical_Properties Value in Import files
Install	Construction
Existing	NA
Remove	Decommission
Retired	NA

The model preprocessor calculates these values and writes them out into import files.

> **Note:** Model Extractors must be modified to not filter out devices in the "Install" state.

Model Builds and Commissioned/Decommissioned Devices

The Commissioning Tool moves devices between "Not Commissioned" and "Commissioned" as well as "Not Decommissioned" and "Decommissioned".

If an operator commissions a device, marked as Construction, a model build will not reset the commissioning state (i.e., Subsequent model builds will not undo changes made by the Commissioning Tool).

Effect of ICP Devices on Network Topology

Devices affect the network's topology as follows:

Device State	Commissioned / Decommissioned	Does Device affect Network Model
Install	Not commissioned	No.
Install	Commissioned	Yes. As normal existing device.
Remove	Not decommissioned	Yes. As normal existing device.
Remove	Decommissioned	No.

ICP Device Symbology

The Viewer will hide certain ICP-marked devices and display certain ICP devices with additional symbology.

Device State	Commissioned / Decommissioned	Default Visibility	Special symbology
Install	Not commissioned	Hidden	Yes.
Install	Commissioned	Visible	Yes
Existing	NA	Visible	No
Remove	Not decommissioned	Visible	Yes
Remove	Decommissioned	Hidden	Yes

Note: See "Understanding the Viewer" in the Oracle Utilities Network Management SystemUser Guide for more information on ICP symbology and how to use the Commissioning Tool.

Troubleshooting Issues with ICP Device Symbology

If you notice that some pending construction and pending decommission objects are missing conditions and are not hiding correctly with the Hide/Display option in the Viewer, you can run DBCleanup with the -fixICP option.

To see the objects that are missing conditions:

```
DBCleanup -fixICP -showMe
```

To add the conditions:

```
DBCleanup -fixICP
```

DDService is required to be running. No services need to be stopped or restarted when using this option.

You can run `DBCleanup -fixICP -skipMBSCheck` if you are only performing this ICP cleanup routine and no other model-related routines.

Note: running `DBCleanup -fixAll` does **NOT** run the `-fixICP` option.

Auto Throw-Over Switch Configuration (ATO)

Oracle Utilities Network Management System supports the modeling and visualization of Auto Throw-Over (ATO) devices. Critical customers such as hospitals, manufacturing, financial and emergency services, require higher level of power quality and reliability. These customers are normally provided with a primary and backup source of power to improve the reliability. Utilities deploy automatic throw over devices to switch the load to backup source when the primary source is not available. Often these devices have automatic restoration feature where the load is fed by the primary source when primary source is energized after an outage.

Model Requirements for ATOs

In order to configure ATOs in the Oracle Utilities Network Management System, the Model Build process needs to know what two devices are controlled by the ATO controller. One device must be identified as the primary or preferred feed, which would be normally closed, and the other device would be the secondary or alternate feed, which would be normally open. These relationships and control behaviors are modeled in the ATO_CONTROLLERS table, as shown below:

Field	Format	Comments
H_CLS	N	Class part of the ATO controller handle. Required.
H_IDX	N	Index part of the ATO controller handle. Required.
PARTITION	N	ATO controller partition.
CONTROL_FUNCTION	V32	ATO control function identifier. Required. Values: • 2dev – 2 ATO Devices and no auto-restore • 2dev_arc – 2 ATO Devices, auto-restore, no momentary on restore operation • 2dev_momentary_arc – 2 ATO Devices, auto-restore, and will create a momentary on restore operation
ATO1_CLS	N	Class part of the handle of the primary ATO device. Required.
ATO1_IDX	N	Index part of the handle of the primary ATO device. Required.
ATO2_CLS	N	Class part of the handle of the secondary ATO device. Optional.
ATO2_IDX	N	Index part of the handle of the secondary ATO device. Optional.
PARAM1	N	Delay (in seconds) until primary ATO device is opened during throwover - Optional.
PARAM2	N	Delay (in seconds) until secondary ATO device is opened during auto-return (ignored by control function "2dev" but column presence is still required) - Optional.

Field	Format	Comments
PARAM3	N	Delay (in seconds) between operating primary and secondary ATO devices. If not configured, there is no delay. -Optional.
BIRTH	D	Birth date of when the object is activated into the model
BIRTH_PATCH	N	Patch which activated this object
DEATH	D	Death date of when the object is de-activated from the model
DEATH_PATCH	N	Patch which de-activated this object
ACTIVE	V1	Active flag

Summary Object Configuration

Summary Objects are objects in one world (for example, Geographic World) that reflect events or conditions in another world (for example, Substation World). For example, a substation fence in the geographic world may display the conditions existing on objects within the substation in the internal world view of the substation (like an outage on a breaker in the substation would be reflected on the fence in the geographic world).

To configure this functionality, you need to configure three areas of the model:

1. Verify that summaryobjects is on the DDService in the `~/etc/system.dat` file.
2. Verify that `nms_srs_rules.sql` has a `config` rule for summaryObject set to "`yes`".
3. Verify that all object classes you wish to have summary events reflected on are in the project condition rules file (for example, substation_fences).
4. Substation fences, when build, must define a location in the `.mb` file. For example:

```
ADD substation_fence 2 {
    LOCATION = <10210.2>;
    ALIAS[OPS] = "SUB_Lake";
    DIAGRAM[1022] (1022) = {
        SYMBOLOGY = 101;
        HEIGHT = 500.000000;
        GEOMETRY = {
            (2270311.397232,460321.122269),
            (2270311.397232,459286.466476),
            (2271217.293103,459286.466476),
            (2271217.293103,460321.122269),
            (2270311.397232,460321.122269)
        };
    };
    ATTRIBUTE[Latitude]="  40.92498";
    ATTRIBUTE[Longitude]=" -81.40776";
};

ADD LOCATION <10210.2> {
    NAME = "SUB_Lake";
    DESC = "Lake Substation";
    REFERENCE = (2270311.397232,460321.122269);
};
```

5. All objects in the substation partition that you want the events and conditions reflected on the substation fence must belong to the same location. For example:

```
ADD rack_circuit_br 1500 {
    PHYSICAL_PROPERTY = SUB;
    VOLTAGE = 13800;
    NCG = 63;
    PHASES = 7;
    LOCATION = <10210.2>;
    PORT_A = <444.2523.2>;
    PORT_B = <444.2522.2>;
    ALIAS[GIS] = "Circuit Breaker.270";
    ALIAS[OPS] = "BR241XFM";
    DIAGRAM[1094] (1094) = {
        RANK = 65544;
        SYMBOLOGY = 10507;
        HEIGHT = 500.000000;
        GEOMETRY = {
            (205.811207,412.902928),
            (205.811207,391.655951)
        };
    };
    ATTRIBUTE[gmd_location] = "Lake Substation";
    ATTRIBUTE[gmd_comment] = "0.0000";
```

To use LatLong to convert coordinates from parameters instead on files, use the `-script` option. Here are examples:

Convert x,y coordinate to lat,long:

```
$  LatLong OH83-NF LL -script 2270808.750654 461097.566511
       -81.40592405         40.92709161
```

Convert long,lat to x,y:

```
$  LatLong LL OH83-NF -script -81.40592405 40.92709161
   2270808.75152434     461097.56670991
```

Adding Latitude and Longitude Attributes to Objects in the Model Build Process

The NMS LatLong tool will populate latitude and longitude attributes to objects in a given model import (.mb) file.

The format of the command is:

```
LatLong {src_proj} {dest_proj} {infile} {outfile} [{precision}]
```

where:

`src_proj` = projection name * (for example, CO-N CO-S GA83-GeorgiaPwr CT MN-S UTM-15N)

`dest_proj` = projection name * (for example, LL)

`infile` = input file name for the `.mb` file

`outfile` = output file name for the `.mb` file

`precision` = optional number of digits to the right of the decimal (def = 8)

Notes:

- See the $OMS_PREFIX/data/MAPPING/coordsys.asc file for supported projections.
- Objects with __GEO_REF_X and __GEO_REF_Y attributes will be used for the coordinates to transform to the dest_proj instead of the GEOMETRY COORDs on the object.

or:

```
LatLong {src_proj} {dest_proj} -script {x or longitude} {y or
latitude} [{precision}]
```

The call to this program is typically added to the `[project]-build-map` script:

```
_echo "Preprocessor complete, doing latlong..." ${logdir}/
${mapPrefix}.log

LatLong \
        OH83-NF LL \
         ${OPERATIONS_MODELS}/patches/${mapPrefix}.mb \
         ${OPERATIONS_MODELS}/patches/${mapPrefix}.mbll \
     >> ${logdir}/${mapPrefix}.log 2>&1

if [[ ! -f ${OPERATIONS_MODELS}/patches/${mapPrefix}.mbll ]]
then
    _echo "WARNING:Unable to add lat/long to MP to ${mapPrefix}.mb"\
      ${logdir}/${mapPrefix}.log
else
    mv ${OPERATIONS_MODELS}/patches/${mapPrefix}.mbll \
       ${OPERATIONS_MODELS}/patches/${mapPrefix}.mb
    _echo "Finished converting MP file $mapPrefix to ${mapPrefix}.mb"\
      ${logdir}/${mapPrefix}.log
fi
```

Coordinate Systems

A coordinate system is a definition of how we numerically describe the position of objects and events. The Network Management System supports two dimensional coordinate systems, typically with an x and y value, sometimes using latitude and longitude.

In the Network Management System, only objects in the same coordinate system can be loaded in the Viewer at any given time.

Geographic Coordinate System

Many Network Management Systems implementations need to be concerned with only one coordinate system: The geographic coordinate system assuming the source of the model data is a Geographic Information System (GIS). Typically the GIS coordinate system is specified in x and y values with units being feet to meters.

In the Network Management System, we specify a coordinate system in the model import files as part of the PARTITION definition using the COORD_SYSTEM value. By convention, the primary GIS coordinate system is identified as COORD_SYSTEM=0.

Quasi-Geographic Coordinate System

Some GIS systems, specifically SmallWorld and Intergraph, have the ability to model a schematically drawn quasi-geographical representation of the model. This representation is often drawn in the same coordinate space as the geographic data but is an alternate representation of the device and conductor layout to be easier to see. These alternate representations are usually orthogonally drawn and devices are spaced out so they can be seen while zoomed out at a higher level.

Typically, by convention, any quasi-geographic coordinate system or systems would be given a COORD_SYSTEM number between 1 and 9. Most projects that implement quasi-geographic coordinate systems will have only one coordinate system, would use COORD_SYSTEM=1.

Internal World Coordinate Systems

Internal worlds are drawings using a simple Cartesian coordinate system with no relationship to geographic coordinate system. Typical, internal worlds are used for complex electrical facilities that occupy a small geographic space. Examples are substations, complex switch cabinets, and low-voltage stations. These drawings are typically drawn orthogonally and convey electrical connectivity orientation as opposed to spatial orientation.

As opposed to Geographic and Quasi-Geographic coordinate systems, which can contain many partitions that can be tiled or laid out together over space, an internal world coordinate system will have one partition per coordinate system. The numbering system for internal world coordinate systems by convention will match the partition number, which start at 1000 and go up from there for each partition in the system.

Overview Schematic Coordinate Systems

The Network Management System can create auto-generated overview schematics as described earlier in this chapter. Typically a Network Management System will configure one schematic set for each major level (*i.e.*, distribution, sub transmission, and transmission). Each schematic set will be given a coordinate system to use for the resulting schematic partitions using the `-coordSystem` parameter. By convention, Network Management Systems implementation will use COORD_SYSTEM number from 100-199 for overview schematics.

The overview schematic coordinate systems should be specified in the viewer configuration to allow loading these views using the schematic focus/target button. This specification should be in the VIEWER_GLOBAL_PROPERTIES.inc file using the viewer.schematic_coord_systems property.

One Line Schematic Coordinate Systems

In addition to overview schematics, the Network Management System supports one line feeder schematics. Feeder one lines schematics can be loaded individually and dynamically you choose tie switches to load electrically adjacent feeders and they are loaded next to the original feeder and the Viewer highlights the common devices between the loaded feeders.

The feeder one line schematic coordinate systems should be specified in the viewer configuration to allow loading these views using the one line schematic focus/target button. This specification should be in the VIEWER_GLOBAL_PROPERTIES.inc file using the viewer.single_circuit_schematic_coord_sys property and defaults to a value of 10.

Latitude and Longitude Coordinate Systems

Many interfaces to the Network Management System require latitude and longitude values. The Viewer has a Coordinate report that displays map coordinates and latitude and longitude values. The Customer Self Service, Operations Mobile Application, and Mobile Workforce Management interface require latitude and longitude values. Longitude and Latitude are either defined as part of the model build process or dynamically calculated.

The model build process used the LatLong tool in addition to the model preprocessor to populate the latitude and longitude values on objects. This process is discussed elsewhere in this document.

The dynamic calculation of latitude and longitude is based on configuration of source coordinate system and target latitude and longitude specification. This configuration is done in the [project]_parameters.sql file using the MBS_GEO_* and MBS_LL_* parameters.

Symbology

The Viewer displays all model objects and conditions as symbols, either vector symbols or raster symbols. This symbology system is made up of four types of symbols (with the indicated symbol identifier (SIN) range):

- Firm Symbols (30,000 - 99,999)
- Hard Symbols (100 - 2100)
- Soft, Pixmap, or Scalable Vector Graphics Symbols (2100 - 29,999)

Firm and Hard symbols are generally used for linear objects like conductors, roads, and boundaries. Soft, pixmap, and SVG symbols are generally used for devices (switches, transformers, shunt devices, etc.) and other "point" devices.

Firm Symbols

Firm symbols have an eight digit SIN based on the pattern: **TTTLSDCC.** Each digit defines an aspect of the 1D symbol that is drawn in the Viewer.

- **TTT:** line thickness. Zero = minimal width (1 pixel). Greater than zero = thickness in world coordinates (feet, meters, etc.)
- **L**: long dash length
- **S**: space length
- **D**: dash pattern
- **CC**: color code

Firm symbols are indicated by SINs ranging from 30000 to 99999.

L - Long Dash Length. The long dash length is the continuous part of the line between the spaces and short dashes, if any. This digit determines how many pixels the long dash will be. It must be 3 or greater to classify as a firm symbol.

S - Space Length. The space length is the gap between long dashes and short dashes. This digit defines the space's pixel length as **2*S**. An S value of zero results in a solid line even when the dash pattern is greater than zero.

S Value	Length (pixels)
0	0 (No Space)
1	2
2	4
3	6
4	8
5	10
6	12
7	14
8	16
9	18

D - Dash Pattern. The short dash pattern defines the number and size of short dashes in the line. There can be from zero to three short dashes in each line pattern. The short dashes can be one pixel points, space sized dashes or double space sized dashes.

D Value	Description	Sketch
0	No short dashes	_____
1	One point, one pixel	_____ . _____
2	Two points, one pixel each	_____ . . _____
3	Three points, one pixel each	_____ . . . _____
4	One short dash, 1 * S	_____ _ _____
5	Two short dashes, each 1 * S	_____ _ _ _____
6	Three short dashes, each 1 * S	_____ _ _ _ _____
7	One short dash, 2 * S	_____ __ _____
8	Two short dashes, each 2 * S	_____ __ __ _____
9	Three short dashes, each 2 * S	_____ __ __ __ _____

CC - Color Code. The line color is specified by a two digit color code.

Code		Name	RGB	Hex
00		black	0/0/0	#000000
01		white	255/255/255	#FFFFFF
02		red	255/0/0	#FF0000
03		yellow	255/255/0	#FFFF00
04		green	0/255/0	#00FF00
05		cyan	0/255/255	#00FFFF
06		blue	0/0/255	#0000FF
07		magenta	255/0/255	#FF00FF
08		orange	255/165/0	#FFA500
09		pink	255/192/203	#FFC0CB
10		tan	210/180/140	#D2B48C
11		gray	190/190/190	#BEBEBE

Code		Name	RGB	Hex
12		navy	0/0/128	#000080
13		brown	165/42/42	#A52A2A
14		purple	160/32/240	#A020F0
15		salmon	250/128/114	#FA8072
16		grey10	26/26/26	#1A1A1A
17		grey20	51/51/51	#333333
18		grey30	77/77/77	#4D4D4D
19		grey40	102/102/102	#666666
20		grey50	127/127/127	#7F7F7F
21		grey60	153/153/153	#999999
22		grey70	179/179/179	#B3B3B3
23		grey80	204/204/204	#CCCCCC
24		grey90	229/229/229	#E5E5E5
25		red1	255/0/0	#FF0000
26		red2	238/0/0	#EE0000
27		red3	205/0/0	#CD0000
28		red4	139/0/0	#8B0000
29		limegreen	50/205/50	#32CD32
30		turquoise	64/224/208	#40E0D0
31		violet	238/130/238	#EE82EE
32		violetred	208/32/144	#D02090
33		deeppink	255/20/147	#FF1493

Code		Name	RGB	Hex
34		aquamarine	127/255/212	#7FFFD4
35		khaki	240/230/140	#F0E68C
36		goldenrod	218/165/32	#DAA520
37		gold	255/215/0	#FFD700
38		coral	255/127/80	#FF7F50
39		maroon	176/48/96	#B03060
40		wheat	245/222/179	#F5DEB3
41		green3	0/205/0	#00CD00
42		green4	0/139/0	#008B00
43		coral2	238/106/80	#EE6A50
44		yellow1	255/255/0	#FFFF00
45		yellow2	238/238/0	#EEEE00
46		blue4	0/0/139	#00008B
47	not used			
48		orange1	255/165/0	#FFA500
49		orange2	238/154/0	#EE9A00
50		brown4	139/35/35	#8B2323
51		magenta1	255/0/255	#FF00FF
52		magenta3	205/0/205	#CD00CD
53		steelblue1	99/184/255	#63B8FF
54		steelblue2	92/172/238	#5CACEE
55		cyan4	0/139/139	#008B8B
56		orange4	139/90/0	#8B5A00

Code	Name	RGB	Hex
57	yellow4	139/139/0	#8B8B00
58	moccasin	255/228/181	#FFE4B5
59	light pink	255/182/193	#FFB6C1
60	deep sky blue	30/144/255	#1E90FF
61	medium aquamarine	102/205/170	#66CDAA
62	snow1	255/250/250	#FFFAFA
63	blue1	0/0/255	#0000FF
64	cadet blue	95/158/160	#5F9EA0
65	dark green	0/100/0	#006400
66	sea green	46/139/87	#2E8B57
67	firebrick	178/34/34	#B22222
68	tomato	255/99/71	#FF6347
69	light goldenrod	238/221/130	#EEDD82
70	goldenrod1	255/193/37	#FFC125
71	hotpink1	255/110/180	#FF6EB4
72	not used		
73	magenta4	139/0/139	#8B008B
74	chocolate4	139/69/19	#8B4513
75	wheat1	255/231/186	#FFE7BA
76	thistle4	139/123/139	#8B7B8B
77	steel blue	70/130/180	#4682B4
78	maroon4	139/28/98	#8B1C62
79	coral1	255/114/86	#FF7256

Code		Name	RGB	Hex
80		deeppink1	255/20/147	#FF1493
81		laurellee	2/175/143	#02AF8F
82		slate grey	112/128/144	#708090
83		royal blue	65/105/225	#4169E1
84		orchid	218/112/214	#DA70D6
85		dark orange	255/140/0	#FF8C00
86	not used			
87				
88				
89				
90				
91				
92				
93				
94				
95				
96				
97				
98				
99		eaudenil	148/218/176	#94DAB0

Hard Symbols

Hard symbols have an eight digit SIN based on the pattern: **TTT0LLCC.**

- **TTT:** line thickness. Zero = minimal width (1 pixel). Greater than zero = thickness in world coordinates (feet, meters, and so forth)
- **LL**: line style
- **CC**: color code

Hard symbols are indicated by SINs ranging from 100 to 2100; if the SIN is less than 1000, assume a zero before the first digit.

LL - Line style. Choose a line style number based on the desired dash pattern and background color. Dash pattern refers to the alternating number of pixels to draw of specified color and background color. The first number draws the prescribed color, CC; the second number draws the background color; the third number, if any, draws the prescribed color and so on.

Line Style Number	Dash Pattern (pixels)	Background Color
1	None	Transparent
11	10,1	Transparent
12	10,1,2,1	Transparent
13	10,1,2,1,2,1	Transparent
14	10,1,2,1,2,1,2,1	Transparent
15	20,10	Grey30
16	50,10,10,10	Grey30
17	75,10,10,10,10,10,10,10	Grey30
18	2,4	Transparent
19	15,15	Black
20	15,15	White

CC - Color Code. The color codes are the same as those listed under firm symbols. Use the color code to prescribe the foreground color of the dash pattern. The SIN 106 is drawn in the Viewer as a solid blue line. The SIN 1614 is drawn in the Viewer as a dashed line with 50 pixels of purple, 10 pixels of gray30, 10 pixels of purple and 10 pixels of gray30.

1D Width Multiplier

The width multiplier increases the thickness of the firm or hard 1D symbol. Add one or more digits ranging from 1 to 29999 to the base SIN to increase the width of the line drawn on the Viewer. The multiplier increases the width of the line proportionally to the map scale so that the line width increases and decreases with zoom level. If no multiplier is specified or if the multiplier is 1, the line width is always one pixel regardless of zoom level. The actual width of the symbol in pixels is calculated at run time. Note that the results of the multiplier vary with each model.

The width multiplier is added to the beginning or left side of the base SIN starting with the sixth digit. Since hard SINs only have four digits, a zero must be added prior to adding the multiplier. For example, 5001324 is a hard symbol with base SIN 1324. The width multiplier is 50. The extra zero is a placeholder only. The firm symbol 5045733 with base SIN 45733 also has a width multiplier of 50. Divide the symbol id number by 100,000 to determine the width multiplier.

Soft Symbol Definitions

Soft symbols are classified as a point or line. There is no graphic editor for the soft symbols, which are considered to be legacy symbols and customers are encouraged to move to SVG symbols. They are indicated by SINs ranging from 2101 to 29999. Symbol definitions, in the [project]_SYMBOLS.sym file, have a regular pattern consisting of a header and a body. The first line of the header is called the *header line* and is followed by additional required key attribute lines.

Symbol Header

Header Line

The first header line begins with `SH` and is then followed by the symbol type, symbol code, and the symbol name:

```
SH [symbol_type] [symbol_code] [symbol_name]
```

- `symbol_type`: a point (**`P`**) or a line (**`L`**)
- `symbol_code`: the unique symbol identification number (**`SIN`**)
- `symbol_name`: a text string that names the symbol.

Examples

- Point transformer with SIN 2200:

  ```
  SH P 2200 xfmr
  ```

- Line switch with SIN 2201:

  ```
  SH L 2201 switch
  ```

Anchor Points: A1 and A2

Anchor points determine the default focus point for line and point symbols.

- The focus point for line symbols is the midpoint between two anchor points (`A1` and `A2`).
- The `A1` anchor point is the default focus point for point symbols.

Once the drawing coordinates are determined, the symbol is scaled and rotated around its focus point.

A1 [x] [y]

A required record that defines the first anchor point of a line symbol or the only anchor point for a point symbol.

Examples

- Add the default focus point for the transformer:

```
SH P 2200 xfmr
A1 0 0
```

- Add the first anchor point for the switch:

```
SH L 2201 switch
A1 -10 0
```

A2 [x] [y]

A required record for line symbols that defines their second anchor point. The second anchor point line has the following format.

```
A2 [x] [y]
```

- Add the second anchor point for the switch with default focus point (0,0):

```
SH L 2201 switch
A1 -10 0
A2 10 0
```

Color Definition

CF [foreground_color_number] [background_color_number]

A required record that defines the colors used for filled objects and double dash lines. The foreground color for filled objects is the line color and the background color is the fill color, but can be configured to use any RGB color.

Examples

- Set the transformer foreground color to 3 (yellow) and background color to 0 (black):

```
SH P 2200 xfmr
A1 0 0
CF 3 0
```

- Set the switch foreground color to 1 (white) and background color to 0 (black):

```
SH L 2201 switch
A1 -10 0
A2 10 0
CF 1 0
```

Symbol Body [SB]

The first line of the symbol body contains only **SB**, which denotes that the symbol header has ended and the symbol body definition is beginning. Each subsequent line in the symbol body defines a new aspect of the soft symbol, including color changes, line style changes, draw actions, and movements.

> **Note:** The end of the symbol body is designated by the end of the file or the beginning of a new symbol.

Example

- Transformer:

```
SH P 2200 xfmr
A1 0 0
CF 3 0
SB
```

- Switch:

```
SH L 2201 switch
A1 -10 0
A2 10 0
CF 1 0
SB
```

The following sections describe valid actions for the symbol body.

PEN Definitions

s [color_number]

Sets the pen to a specified color that cannot be overridden by the Viewer selection color. If a symbol drawn with this pen is selected in the Viewer, it does not blink or change colors.

Example

- Set the switch pen color to black:

```
SH L 2201 switch
A1 -10 0
A2 10 0
CF 1 0
SB
s 100
```

SO [color_number]

Sets the pen to a specified color that can be overridden by the Viewer selection color. If a symbol drawn with this pen is selected in the Viewer, it blinks or changes color.

Example

- Set the transformer pen color to black:

```
SH P 2200 xfmr
A1 0 0
CF 3 0
SB
SO 100
```

Line

W [width]

Specifies the line width. The results of this value varies for each model.

Example

- Set the switch's line width to 1:

```
SH L 2201 switch
A1 -10 0
A2 10 0
CF 1 0
SB
s 100
W 1
```

L [line_style_number] [length] [length] [length]...

Sets the line style and dash pattern.

Valid values for `[line_style_number]` are:

- 1 - solid
- 2 - dash; alternate between specified color and transparent
- 3 - double dash; alternate between foreground color and background color

The `[length]` parameters are optional. They specify the segment lengths for the dash pattern. There can be many `[length]` parameters, but the last one must be equal to zero.

Example

- Set the switch's line style to 1 (solid):

```
SH L 2201 switch
A1 -10 0
A2 10 0
CF 1 0
SB
s 100
W1
L 1
```

D [x1] [y1] [x2] [y2]

Draws a line symbol between two points (x1, y1) and (x2, y2).

Example

- Draw a solid line with a width of 1 starting at (0,0) ending at (6,0) for the switch:

```
s 100
W1
L1
D 0.0 0.0 6.0 0.0
```

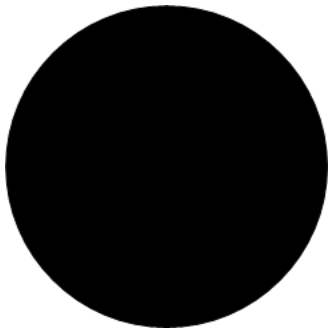
CIRCLE

C [x] [y] [radius]

Draws a filled circle with center (x, y) and a specified radius.

- Draw a black filled circle at (0,0) with radius 2.5:

```
C 0 0 2.5
```



- Draw a black filled circle at (0,0) with radius .3:

```
s 100
W1
L1
D 0.0 0.0 6.0 0.0
C 0.0 0.0 .3
```

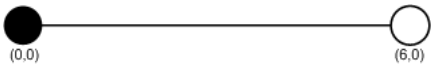
c [x] [y] [radius]

Draws an open circle with center (x, y) and a specified radius.

Example

- Draw an open circle with center (6,0) and radius .3:

```
s 100
W1
L1
D 0.0 0.0 6.0 0.0
C 0.0 0.0 .3
c 6.0 0.0 .3
```



BOX

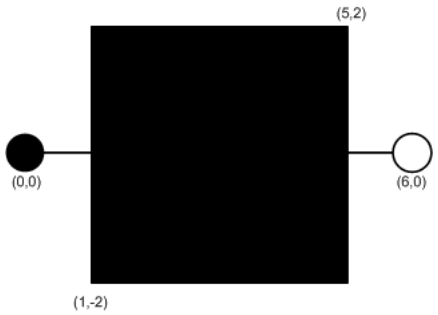
B [x1] [y1] [x2] [y2] [angle]

Draws a filled box between the opposite corners, (x1, y1) and (x2, y2), with the specified angle of rotation.

Example

- Draw a black filled box

```
s 100
W1
L1
D 0.0 0.0 6.0 0.0
C 0.0 0.0 .3
c 6.0 0.0 .3
B 1.0 -2.0 5.0 2.0 0.0
```



b [x1] [y1] [x2] [y2] [angle]

Draws an open box between the opposite corners, (x1, y1) and (x2, y2), with the specified angle of rotation.

Text

t [height] [width] [vertical_justification] [horizontal_justification]

Sets the height and width of the text at a specified justification. Vertical and horizontal justification have the following values:

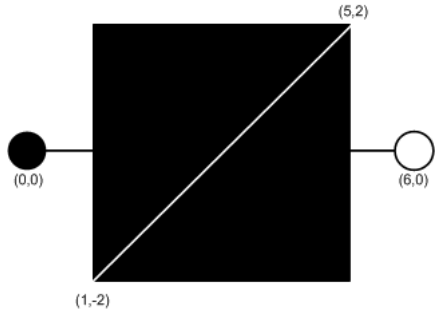
- 0 - left or bottom
- 1 - center
- 2 - right or top

The text is drawn with the 'T' record, but the 't' record must be defined first.

Example

Draw a diagonal line with a white pen color; define text attributes with vertical justification = 0 and horizontal justification = 0

```
s 1
D 1.0 -2.0 5.0 2.0
t 1.0 1.0 0 0
```



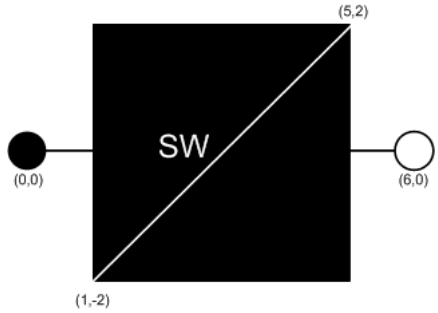
T [x] [y] [angle] "[string]"

Draws the text, "[string]", at (x, y). The text formatting is defined by the 't' record and must be defined prior to the 'T' record.

Example

- Draw the text "SW" at (2,0) with specified text attributes.

```
T 2.0 0.0 0.0 "SW"
```



Polygon

M [x] [y]

Defines the first coordinate for a filled polygon. This record must precede the 'P' action.

Example

- Set the pen color to grey70 and define the first point of the polygon for the transformer:

```
s 22
M 0.0 2.0
```

P [x] [y]

Defines the next coordinate for a filled polygon. Use this action to specify as many points as necessary. This record follows the ‘M’ action and precedes the ‘F’ action.

Example

- Set the remaining points of the polygon for the transformer:

```
P -1.7 -1.0
P 1.7 -1.0
```

F [x] [y]

Defines the last coordinate for a filled polygon. This record follows the ‘P’ action and is the same as the ‘M’ action. It finishes and fills the polygon.

Example

- Finish and fill the polygon. The result is the transformer symbol, xfmr.

```
F 0.0 2.0
```

ARC

a [x] [y] [radius] [begin angle] [end angle]

Draws a circular arc at (x, y) with radius from begin angle to end angle.

SCALED OBJECTS (line, circle, box, polygon)

SW [w]

Defines the scaled line width as a percentage of the distance between anchor points.

N

No scale option for lines, circles, boxes or polygons. This must be defined on the same line as the object this record applies to.

Z [A1] or [A2] or [x][y]

Overrides the default focus point of a line, circle, box or polygon. This must be defined on the same line as the scaled object this record applies to.

Hover Text

H "[string]"

Adds a tooltip that is activated when the user's mouse hovers over the symbol.

Example

- Add "Probable Service Outage" to the probable service outage (PSO) condition symbol.

```
SH P 4001 probable-service-outage
C#
A1 0 0
CF 1 0
SB
s 4
C 0 0 70
SO 1
B -47.36 47.36 47.36 -47.36 45
s 100
t 80 40 1 1
T 0 0 0 "PSO"
H "Probable Service Outage"
```

Pixmap Symbols

Use $NMS_CONFIG/jconfig/ops/viewer/properties/RasterSymbols.properties to specify the image file to use for a given symbol. For example:

```
# This contains a mapping of symbols that should be
# displayed as raster images
# The first file is the normal image. If a second image is listed,
# it is for the selected image. example:
#14042=sym_green_truck.gif,sym_green_truck_sel.gif

#14042=sym_crew.gif
#14043=sym_red_green_truck.gif
#14044=sym_orange_truck.gif
```

SVG Symbols

SVG symbols offer more complicated device, condition, and outage symbols than the standard `.sym` symbols, and can be edited and enhanced more easily using standard tools.

> **Note:** The examples in this section use the Inkscape Editor, which is available from inkscape.org.

To use SVG symbols, add your `.svg` symbols to your `$NMS_CONFIG/jconfig/ops/viewer/images` directory and add references to them to the `$NMS_CONFIG/jconfig/ops/viewer/properties/SVGSymbols.properties` file.

The `SVGSymbols.properties` file maps the `SYMBOLOGY_STATE.symbology_id`, `CONDITION_RULES.symbol_num`, `ANALOG_RULES.symbol_num`, or `QUALITY_RULES.symbol_num` database symbol numbers to the `.svg` symbol file. If you use a selection symbol, the symbol will define the complete visualization of the selected symbol and the viewer.selected_color value will have no impact on the selection symbol. For example:

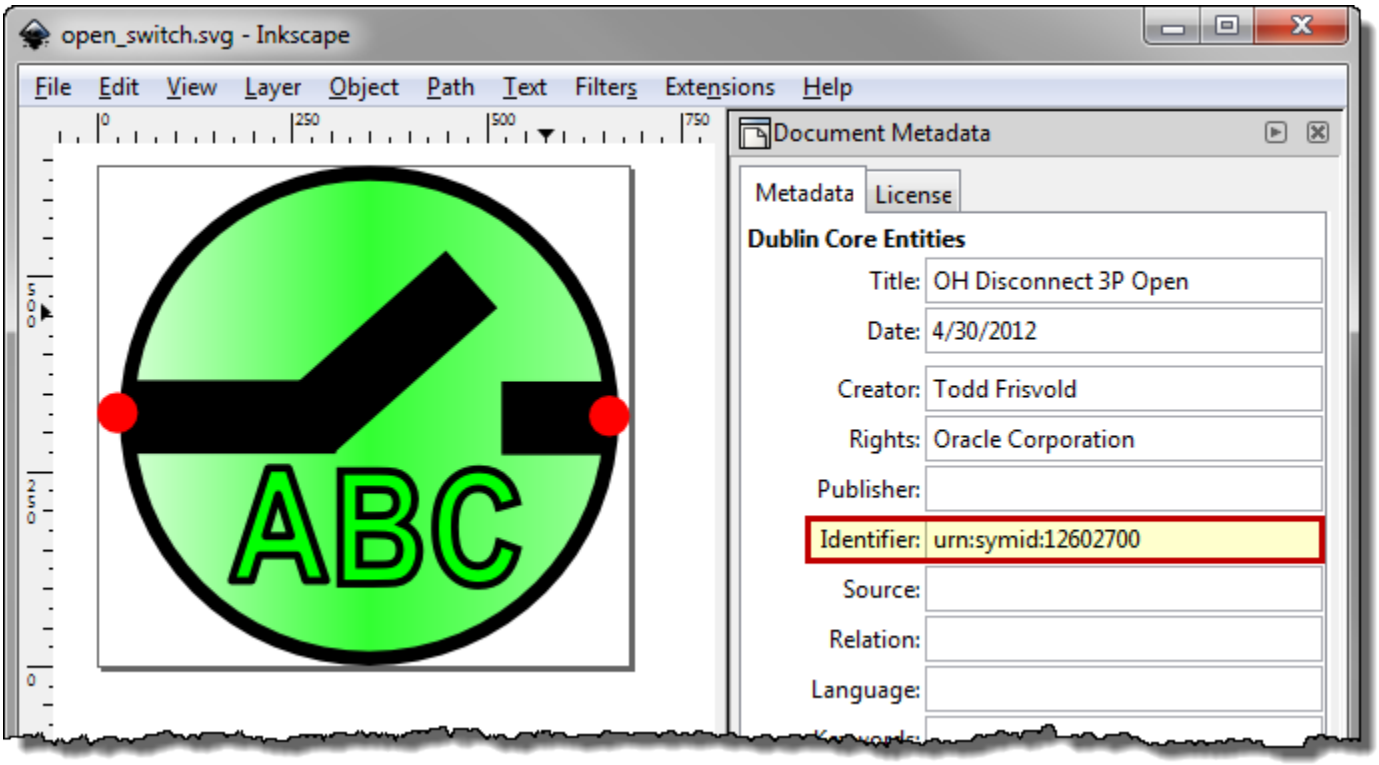
```
4002=Probable_Device_Outage.svg
4004=Real_Device_Outage.svg
14802404=closed_C_fuse.svg
12602700=open_switch.svg
```

SVG symbols have a default selection capability in the Web Viewer which highlights the symbol when selected using the VIEWER_GLOBAL_PROPERTIES.inc file viewer.selected_color StringProperty, which is configured to be "orange" in product. You can specify a specific selection symbol to use for any given symbol by adding a comma and then the selected symbol name to the properties record. For example:

```
14802404=closed_C_fuse.svg,closed_C_fuse_sel.svg
```

You can optionally use the `nms-svg-populate-properties` script to automatically add records to the `SVGSymbols.properties` file. The `.sym` files will need to contain a

special string in the Identifier metadata of the `.sym` file (found in the Document Metadata Identifier field in Inkscape):

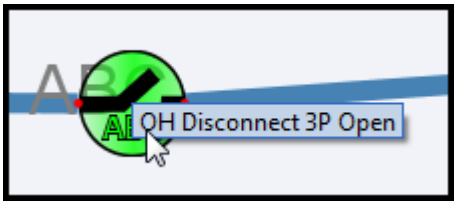


To scan the `.svg` files in the `$NMS_CONFIG/jconfig/ops/viewer/images` directory and update the `SVGSymbols.properties` file, perform the following commands:

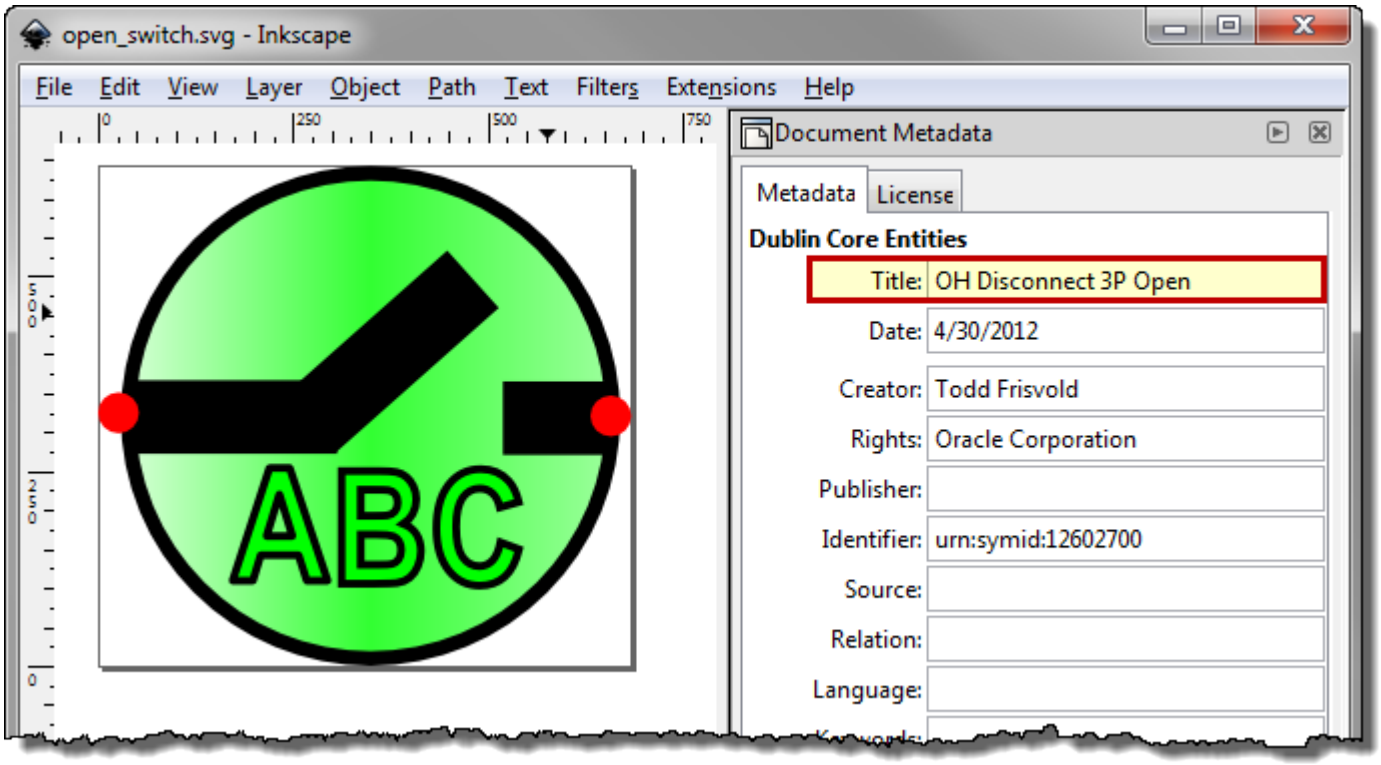
```
$ cd $NMS_CONFIG/jconfig/ops/viewer/images
$ nms-svg-populate-properties
```

The script will scan the `.svg` files in the current directory/path, remove any previously auto-generated records in the `SVGSymbols.properties` file, and add in all new records found in the scan. It will use the Identifier attribute with a string in this format: `urn:symid: iiiiiiiiii`, where `iiiiiiiiii` is the symbology Id to associate with this symbol.

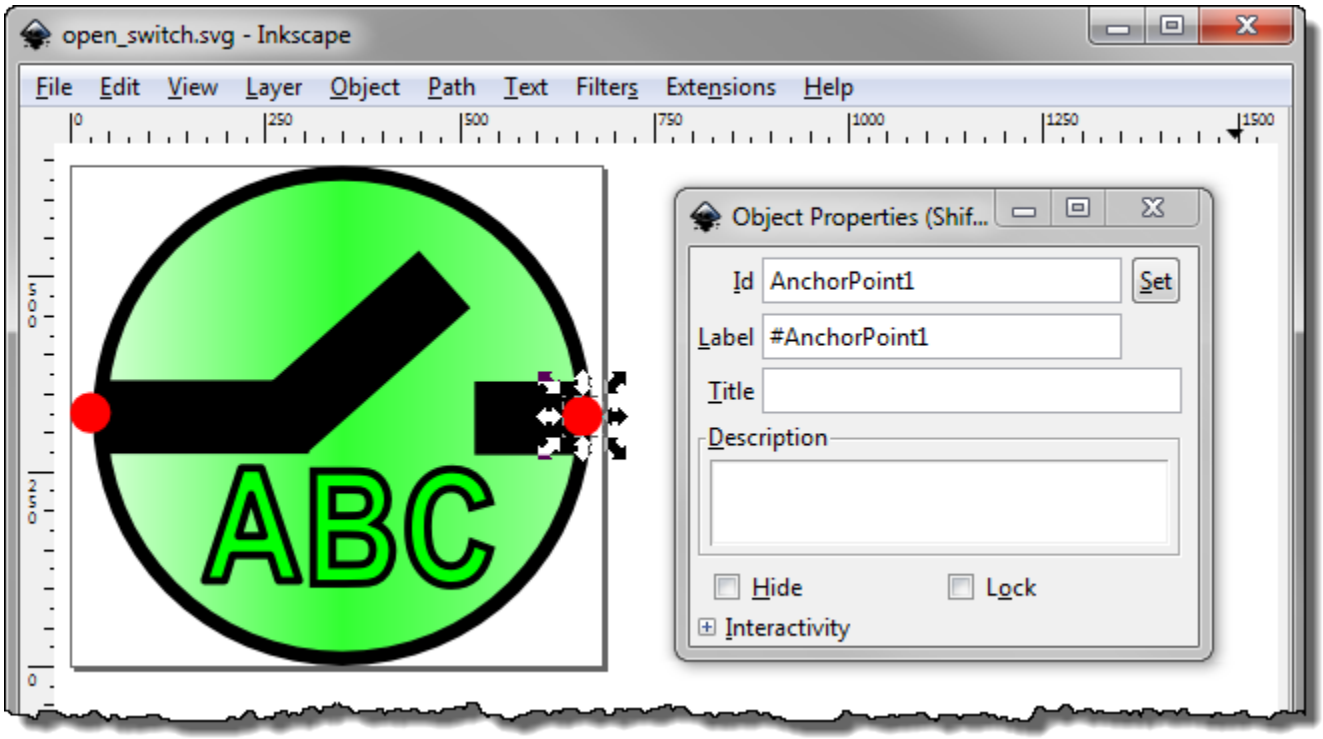
The SVG `[title]` element is used for hover text when the symbol is visualized in the Viewer.



In InkScape, the title is specified in the Document Metadata panel's **Title** field:



To specify Anchor Points in the `.svg` file, place a graphic object and use the `Id` attribute in the Object Properties. Give the anchor points an id value of `AnchorPoint1` or `AnchorPoint2`:



Anchor points can be optionally hidden when drawn in the Viewer by using the Object Fill and Stroke panel and setting the Fill and Stroke Paint Opacity levels to zero. Do not set the Object Properties Hide check box, this will hide the anchor points from the viewer all together and it will not know the anchor points even exist.

For condition symbols, SVG symbols work well if you follow these rules:

1. Use AnchorPoint1 and AnchorPoint2 objects in the symbol to define the vertical spacing desired for the symbol.
2. Use a ConditionPoint object to define the symbol placement and the selection point for the condition. This object will define where the symbol will be place in relation to the end of the leader line drawn from the object to the condition symbol. If you do not place a ConditionPoint, unpredictable symbol placement will occur.
3. If AnchorPoint1 and AnchorPoint2 are not defined, the upper left corner of the symbol will be the default location of AnchorPoint1 and lower right corner of the symbol will be the default location of AnchorPoint2.

Adding Text to SVG Symbols with Inkscape

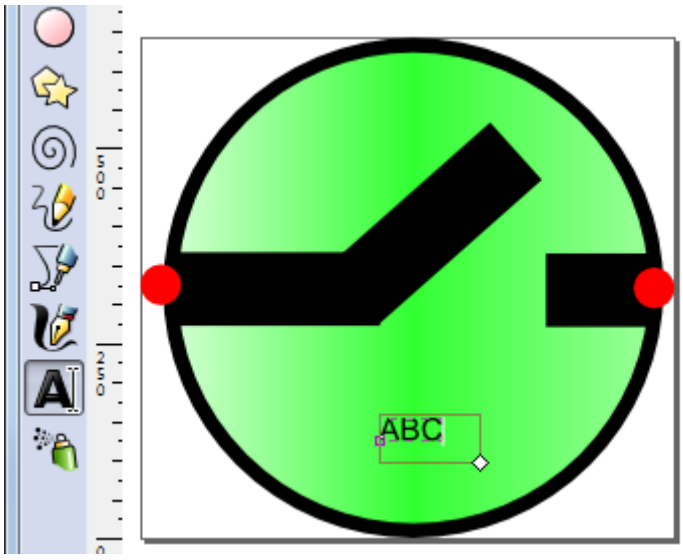
Natively, Inkscape's text tool adds a non-standard SVG element that is not supported by the Viewer (or any standard SVG viewer). Therefore, when using Inkscape to create SVG symbols with text, you must convert the text with Inkscapes text conversion option.

The following example demonstrates the process of adding text, converting the text to a standard SVG element, and validating the file using the Apache Batik SVG Toolkit's Squiggle viewer.

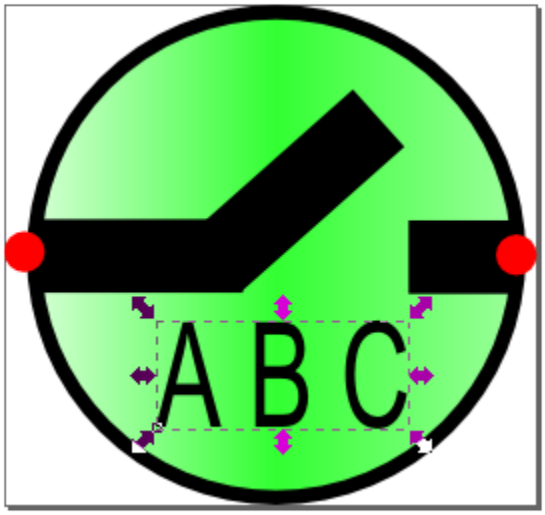
> **Note:** Apache Batik SVG Toolkit is available from the Apache Software Foundation at http://xmlgraphics.apache.org/batik/. The Batik download contains Java

Example: Convert Inkscape text to a standard SVG element and validate with Squiggle.

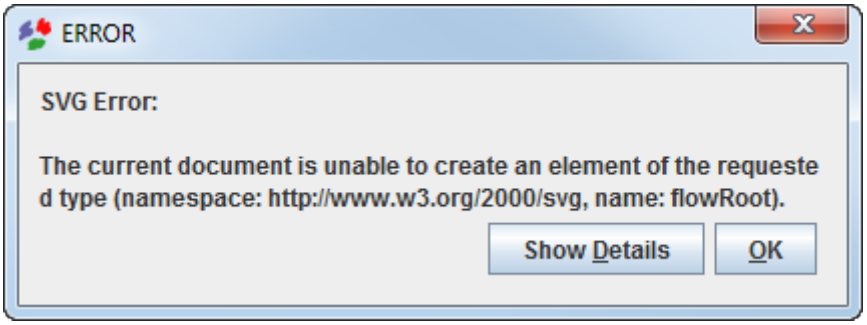
1. Open Inkscape and create a symbol.
2. Add text to the symbol using the **Text** tool.



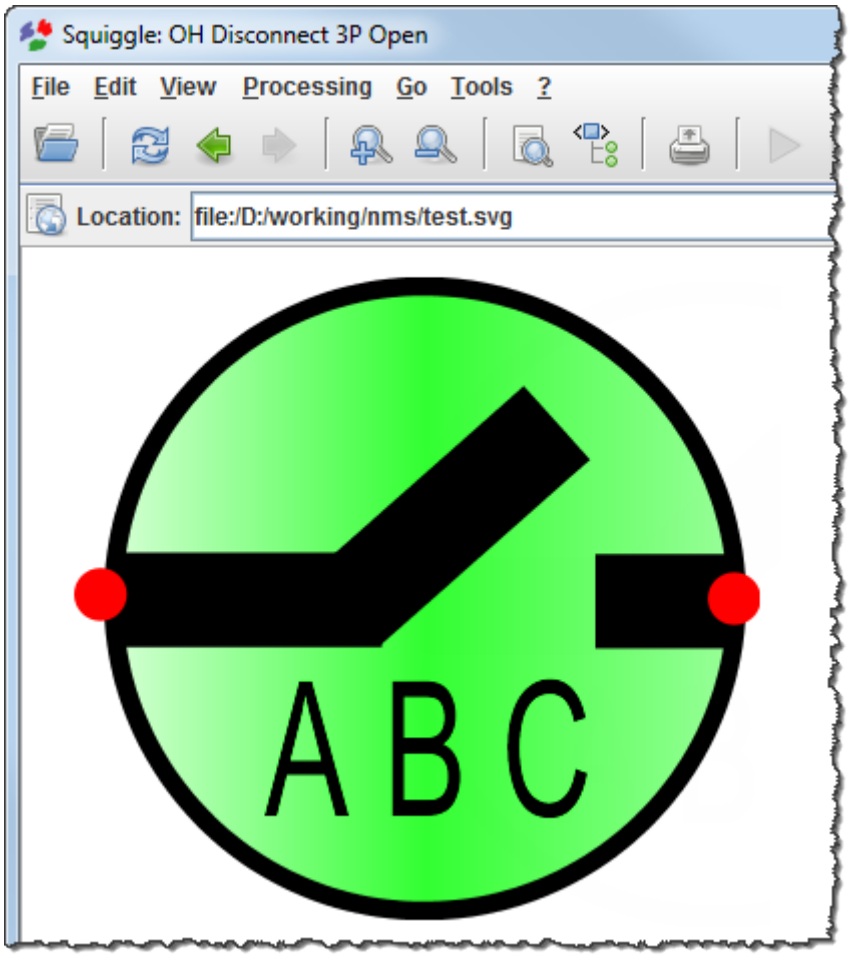
3. Resize and position the text, as needed.



Note: With the non-standard text present, if you save and attempt to open the file with Squiggle, you will receive an error:



4. With the text selected, select **Convert to Text** from the **Text** menu.
5. Save the file.
6. Open the file in Squiggle to validate the SVG is a well-formed XML document.


Squiggle: OH Disconnect 3P Open
File Edit View Processing Go Tools ?
Location: file:/D:/working/nms/test.svg
A B C

SVG Master Symbol Process

The SVG Master Symbol process generates all required variations (for example, open, closed) of a switchable device symbol from one master symbol definition. The master symbol's metadata contains the symbol variation variables that are passed to a script that generates the permutations (`svg-symbol-generator`). The process is automated with a script that reads the master symbol directory and generates all symbol permutations (`autoSvgGenerator`).

autoSvgGenerator Script

The `autoSvgGenerator` script generates SVG symbols from master symbol files located in the `$NMS_CONFIG/jconfig/ops/viewer/images/master` directory. The generated `.svg` files are saved to the `$NMS_CONFIG/jconfig/ops/viewer/images/auto-gen` directory in sub-directories for each class name (for example, fuses, switches).

> **Note:** the auto-gen directory is deleted before processing the master files.

The master symbol `.svg` file requires the object IDs set as specified by the `svg-symbol-generator -help` information and an additional text string in the master `.svg` metadata description attribute with the following format:

```
AutoConfig:[classNumber]:[className]:[tensDigit][:[normalOpenTensDigit]]
```

These values are passed to the respective `svg-symbol-generator` command parameters (see svg-symbol-generator Script on page 8-119).

If the `<normallyOpenTensDigit>` is specified, `autoSvgGenerator` executes `svg-symbol-generator` a second time using the `[normalOpenTensDigit]` value rather than the `-tens` parameter and, optionally, the `-normalopen` parameter. This will use the inclusion of the symbols object with ObjectID = normal_open_indicator and mixed_status_indicator.

If the `-normalopen` parameter is not specified on the `svg-symbol-generator`, the ObjectID = normal_open_indicator and mixed_status_indicator will not be included in the resulting symbols.

For example, an overhead fuse (class name: fuse_oh/class number: 148) with a normally open tens digit of 5, would be defined as:

```
<dc:description>AutoConfig:148:fuse_oh:0:5</dc:description>
```

svg-symbol-generator Script

This script will read a single master file and generate the permutations. Format of the command is:

```
svg-symbol-generator
```

Command parameters:

```
-classname <name>       *REQUIRED*
-classnumber <number>   *REQUIRED, integer greater than zero*
-mastersvg <filename>   *REQUIRED*
-ganged                 *Optional*
-tens <digit>           *Optional Value 0-9, default: 0*
-closedcolor <number>   *Optional Value 000000-ffffff hex,
                         default: e90000*
-opencolor <number>     *Optional Value 000000-ffffff hex,
                         default: 00ff00*
-abnormalcolor <number> *Optional Value 000000-ffffff hex,
                         default: ffff00*
-normalopen             *Optional
```

Master Symbol Specification

The master symbol contains all the symbol objects for all permutations of the symbol. Each object that might have its visibility or color manipulated based on phases and status will be given an object ID based on the following:

The following three IDs will place the object based on phase and will color them black if the status is open or closed on all phases and will color them opencolor/closedcolor if the status is mixed

- ID on A phase text: phase_a_letter
- ID on B phase text: phase_b_letter
- ID on C phase text: phase_c_letter

The following three IDs will place the object based on phase and will color them opencolor/closedcolor

- ID on A phase text: phase_a_letter_status
- ID on B phase text: phase_b_letter_status
- ID on C phase text: phase_c_letter_status

The following three IDs will place the object based on phase and will not color them based on status

- ID on A phase text: phase_a_letter_fixed
- ID on B phase text: phase_b_letter_fixed
- ID on C phase text: phase_c_letter_fixed

The following three IDs will place the object based on phase and will not color them based on status

- ID on A phase line: phase_a_line
- ID on B phase line: phase_b_line
- ID on C phase line: phase_c_line

The following three IDs will place the object based on phase and will color them opencolor/closedcolor

- ID on A phase line: phase_a_line_status
- ID on B phase line: phase_b_line_status
- ID on C phase line: phase_c_line_status

The following three IDs will place the object based on phase and will not color them based on status

- ID on A phase filled: phase_a_filled
- ID on B phase filled: phase_b_filled
- ID on C phase filled: phase_c_filled

The following three IDs will place the object based on phase and will color them opencolor/closedcolor

- ID on A phase filled: phase_a_filled_status
- ID on B phase filled: phase_b_filled_status
- ID on C phase filled: phase_c_filled_status

The following ID will color the gradient object based on status

- ID on gradient shaded status shape: status_color_object

The following ID will color the filled object based on status

- ID on simple filled status shape or group: filled_status_color_object

The following ID will color the line object based on status

- ID on simple line status shape or group: line_status_color_object

The following ID will place the object if the status is abnormal

- ID on abnormal ring object: abnormal_object

The following ID will place the object if it is normally open

- ID on normal open object: normal_open_indicator

The following ID will place the object if its status is mixed

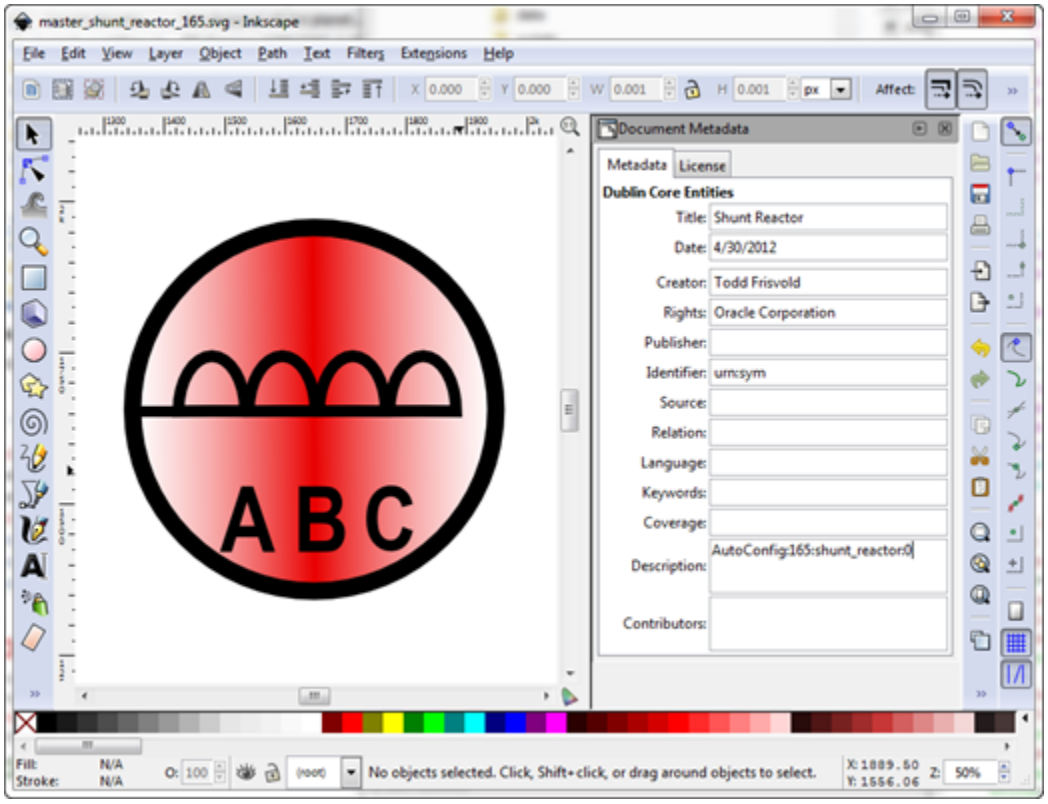
- ID on mixed status object: mixed_status_indicator

The following IDs two will identify the anchor points if the device is open

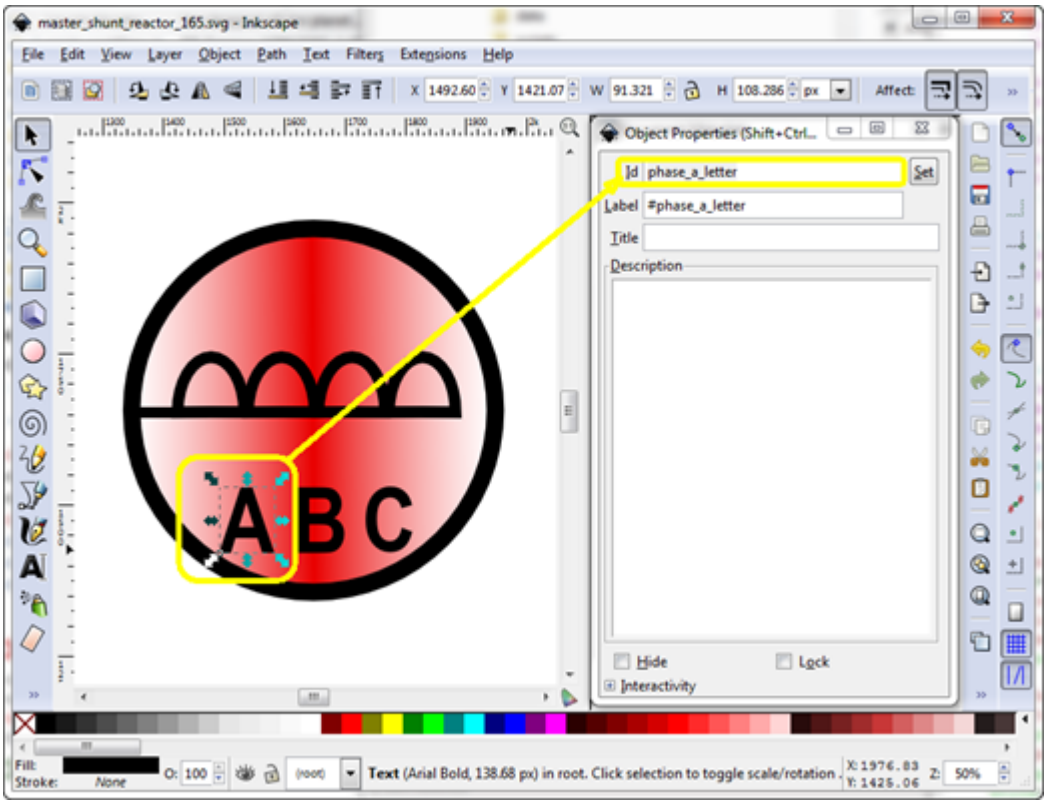
- ID on open anchor point 1 object: anchor_1_if_open
- ID on open anchor point 2 object: anchor_2_if_open

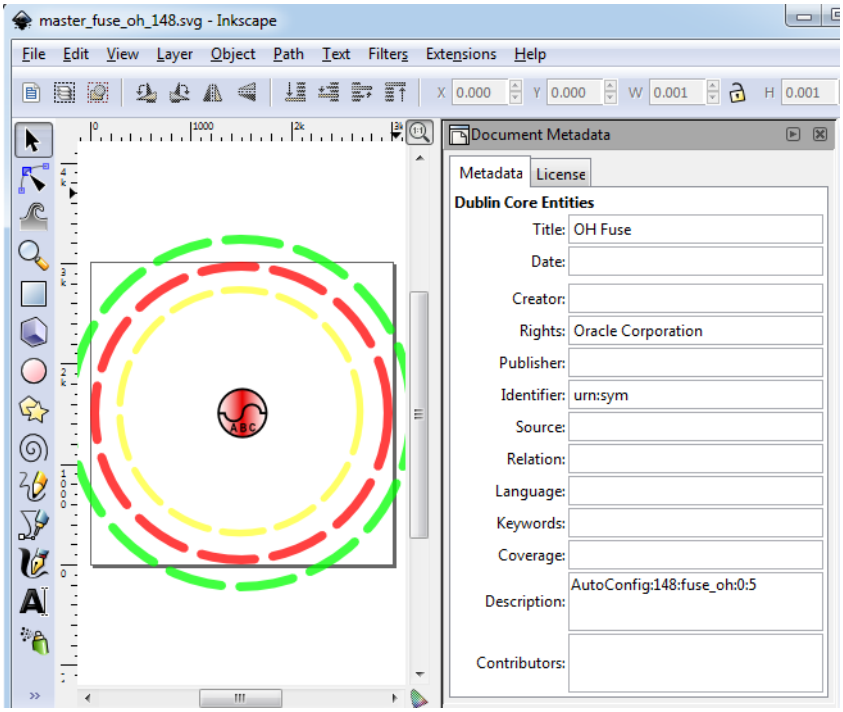
Example

Example Metadata:



Example: setting the A Phase object ID





Example running the script:

```
nmsadmin@nms-vm> cd $NMS_CONFIG/jconfig/ops/viewer/images/master
nmsadmin@nms-vm> ls
master_shunt_reactor_165.svg
nmsadmin@nms-vm> autoSvgGenerator
Info:  Generating symbol shunt_reactor number 0
nmsadmin@nms-vm> ls ../auto_gen/shunt_reactor
shunt_reactor_16502100.svg  shunt_reactor_16512100.svg
shunt_reactor_16502101.svg  shunt_reactor_16512101.svg
shunt_reactor_16502200.svg  shunt_reactor_16512200.svg
shunt_reactor_16502202.svg  shunt_reactor_16512202.svg
shunt_reactor_16502300.svg  shunt_reactor_16512300.svg
shunt_reactor_16502301.svg  shunt_reactor_16512301.svg
shunt_reactor_16502302.svg  shunt_reactor_16512302.svg
shunt_reactor_16502303.svg  shunt_reactor_16512303.svg
shunt_reactor_16502400.svg  shunt_reactor_16512400.svg
shunt_reactor_16502404.svg  shunt_reactor_16512404.svg
shunt_reactor_16502500.svg  shunt_reactor_16512500.svg
shunt_reactor_16502501.svg  shunt_reactor_16512501.svg
shunt_reactor_16502504.svg  shunt_reactor_16512504.svg
shunt_reactor_16502505.svg  shunt_reactor_16512505.svg
shunt_reactor_16502600.svg  shunt_reactor_16512600.svg
shunt_reactor_16502602.svg  shunt_reactor_16512602.svg
shunt_reactor_16502604.svg  shunt_reactor_16512604.svg
shunt_reactor_16502606.svg  shunt_reactor_16512606.svg
shunt_reactor_16502700.svg  shunt_reactor_16512700.svg
shunt_reactor_16502701.svg  shunt_reactor_16512701.svg
shunt_reactor_16502702.svg  shunt_reactor_16512702.svg
shunt_reactor_16502703.svg  shunt_reactor_16512703.svg
shunt_reactor_16502704.svg  shunt_reactor_16512704.svg
shunt_reactor_16502705.svg  shunt_reactor_16512705.svg
shunt_reactor_16502706.svg  shunt_reactor_16512706.svg
shunt_reactor_16502707.svg  shunt_reactor_16512707.svg
```

Converting SYM Files to SVG

The `nms-sym-to-svg` script converts `.sym` files to `.svg` files. The script will create one `.svg` file for each `.sym` symbology preserving the symbology name, ID, anchor points, and all graphic components.

Usage: `nms-sym-to-svg [sym_file] [hideanchors] [createselectionsymbols [colornum]] [oversizecanvas [percentage]] [[symId]]`

where:

- **`hideanchors`** will cause anchor points to be hidden in the resulting SVG files.
- **`createselectionsymbols`** will cause the creation of a selection SVG files in addition to normal symbol SVG files using `[colornum]` as the selection color, typically `[colornum]` value is 8 for orange, see firm symbol colors (0-99) for color code values.
- **`oversizecanvas`** will cause the created svg symbol to have a canvas size `[percentage]` bigger than the stroke size of the symbol. `[percentage]` defaults to 10.0 (10%) and must be greater than zero.
- **`symId`** will be a symbol ID from the `.sym` file to display

To convert your sym files, perform the following steps:

1. Run `nms-sym-to-svg`. The script will output the `.svg` files in the directory where it is run.

   Note: Most projects will want to include the **`hideanchors`** option on the command. If you want to retain the legacy selection symbology, run `nms-sym-to-svg` a second time with the **`createselectionsymbols`** option; otherwise your selection symb will use the auto-selection symbology.

2. Move the svg files to the `$NMS_CONFIG/jconfig/ops/viewer/images` directory
3. Run `nms-svg-populate-properties` to populate the `SVGSymbols.properties` file.

   Note: you need to be in the `$NMS_CONFIG/jconfig/ops/viewer/images` directory to successfully run the `nms-svg-populate-properties` script.

4. Run `nms-install-config --java` to install the new symbols.

SVG Performance Tool

SVG symbols open up many possibilities to symbol creativity however, complex symbols can cause NMS Viewer performance issues too. There are two tools to help the project team to measure SVG symbol rendering performance:

- **svgTestSymbolPerformance:** Measures a single SVG file for rendering performance at different pixel sizes.
- **svgTestSymbolPerformanceReport:** Runs `svgTestSymbolPerformance` and formats output in `.csv` format for import into a spreadsheet.

Both tools require a display to be available to render the symbols, please specify a DISPLAY location before running the commands.

The `svgTestSymbolPerformance` script has the following parameters:

Parameter	Description
`-file [filename]`	svg file to analyze. Required.
`-min [n]`	minimum number of pixels to draw the svg symbol Default = 16
`-max [n]`	maximum number of pixels to draw the svg symbol Default = 1024
`-steps [n]`	number of steps between min and max inclusive to draw the symbol Default = 4
`-draws [n]`	number of draws at each step, Default = 20

Example

Enter the following commands:

```
$ export DISPLAY=host:0.0
$ svgTestSymbolPerformance -file DA_Symbol_14219.svg
```

The system will output:

```
File: DA_Symbol_14219.svg Pixels: 16 WallTime: 0.128 CPUTime: 0.11
Draws: 20
File: DA_Symbol_14219.svg Pixels: 352 WallTime: 0.776 CPUTime: 0.54
Draws: 20
File: DA_Symbol_14219.svg Pixels: 688 WallTime: 2.569 CPUTime: 1.14
Draws: 20
File: DA_Symbol_14219.svg Pixels: 1024 WallTime: 6.641 CPUTime: 2.34
Draws: 20
```

The `svgTestSymbolPerformanceReport` has the following parameters:

```
Infile|directory|file.svg
   - If Infile is specified, the file is scanned for
        svgTestSymbolPerformance report records and reformats them
        into .csv format
   - If directory is specified, the directory is scanned for .svg
        files and runs svgTestSymbolPerformance on each of the .svg
        files found and then reformats the output into .csv format
```

```
   - If an .svg is specified, the file is run into
        svgTestSymbolPerformance and the output is reformatted
        into .csv format
   [Optional parameters]
   - Any additional parameters are passed to the
        svgTestSymbolPerformance process
```

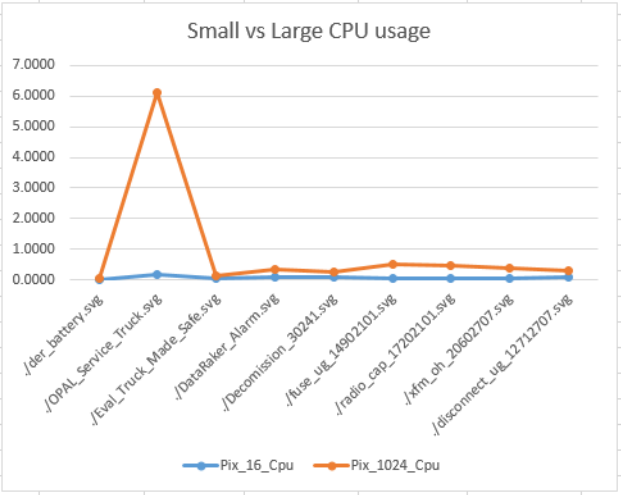
Example

Enter the following commands:

```
$ export DISPLAY=host:0.0
$ ls

DataRaker_Alarm.svg       disconnect_ug_12712707.svg
OPAL_Service_Truck.svg    Decomission_30241.svg
Eval_Truck_Made_Safe.svg  radio_cap_17202101.svg
der_battery.svg           fuse_ug_14902101.svg
xfm_oh_20602707.svg

$ svgTestSymbolPerformanceReport . > report.csv

Processing ./der_battery.svg...
Processing ./OPAL_Service_Truck.svg...
Processing ./Eval_Truck_Made_Safe.svg...
Processing ./DataRaker_Alarm.svg...
Processing ./Decomission_30241.svg...
Processing ./fuse_ug_14902101.svg...
Processing ./radio_cap_17202101.svg...
Processing ./xfm_oh_20602707.svg...
Processing ./disconnect_ug_12712707.svg...

$ cat report.csv

FileName,DrawCount,Pix_16_Wall,Pix_16_Cpu,Pix_352_Wall,Pix_352_Cpu,Pix
_688_Wall,Pix_688_Cpu,Pix_1024_Wall,Pix_1024_Cpu
./der_battery.svg,20,0.023,0.02,0.072,0.05,0.065,0.04,0.046,0.02
./OPAL_Service_Truck.svg,20,0.21,0.18,4.166,4.03,5.207,5.04,6.235,5.93
./
Eval_Truck_Made_Safe.svg,20,0.069,0.06,0.092,0.08,0.099,0.08,0.111,0.0
8
./DataRaker_Alarm.svg,20,0.1,0.09,0.277,0.14,0.4,0.12,0.768,0.25
./Decomission_30241.svg,20,0.075,0.07,0.181,0.11,0.279,0.1,0.495,0.18
./fuse_ug_14902101.svg,20,0.061,0.04,0.33,0.16,0.715,0.24,1.491,0.48
./radio_cap_17202101.svg,20,0.063,0.06,0.296,0.15,0.803,0.25,1.48,0.39
./xfm_oh_20602707.svg,20,0.048,0.05,0.248,0.13,0.502,0.17,0.971,0.32
./
disconnect_ug_12712707.svg,20,0.072,0.07,0.257,0.14,0.451,0.17,0.806,0
.22
```

When brought into a spreadsheet, the analysis can be interesting:

	A	B	C	D	E	F	G	H	I	J
1	FileName	DrawCount	Pix_16_Wall	Pix_16_Cpu	Pix_352_Wall	Pix_352_Cpu	Pix_688_Wall	Pix_688_Cpu	Pix_1024_Wall	Pix_1024_Cpu
2	./der_battery.svg	20	0.0230	0.0200	0.0720	0.0500	0.0650	0.0400	0.0460	0.0200
3	./OPAL_Service_Truck.svg	20	0.2100	0.1800	4.1660	4.0300	5.2070	5.0400	6.2350	5.9300
4	./Eval_Truck_Made_Safe.svg	20	0.0690	0.0600	0.0920	0.0800	0.0990	0.0800	0.1110	0.0800
5	./DataRaker_Alarm.svg	20	0.1000	0.0900	0.2770	0.1400	0.4000	0.1200	0.7680	0.2500
6	./Decomission_30241.svg	20	0.0750	0.0700	0.1810	0.1100	0.2790	0.1000	0.4950	0.1800
7	./fuse_ug_14902101.svg	20	0.0610	0.0400	0.3300	0.1600	0.7150	0.2400	1.4910	0.4800
8	./radio_cap_17202101.svg	20	0.0630	0.0600	0.2960	0.1500	0.8030	0.2500	1.4800	0.3900
9	./xfm_oh_20602707.svg	20	0.0480	0.0500	0.2480	0.1300	0.5020	0.1700	0.9710	0.3200
10	./disconnect_ug_12712707.svg	20	0.0720	0.0700	0.2570	0.1400	0.4510	0.1700	0.8060	0.2200



We can clearly see the OPAL_Service_Truck takes a long time to render in larger pixel renderings. This can cause issues with high-resolution displays, such as 4K monitors. Consider simplifying this symbol to minimize any performance impacts.

Updating Symbology

After the symbology file has been updated and the svg files have been installed, the following command will update the java application server:

```
Action any.publisher* ejb reload_symbology
```

Any running clients will need to be restarted to pick up the new symbology.

Symbology Mapping

The SYMBOLOGY_STATE Table

The SYMBOLOGY_STATE database table contains the mapping from each device's symbology class, state, and off-nominal status to the symbol used for it.

Column	Description
context	At this time, always 'RT_TOPOLOGY'.
symb_state_class	The symbology class, as defined in the DIAGRAM_OBJECTS table.
state	The bitwise switch status, from 0 to 15, or the conductor energization status, from 0 to 23. The conductor energization statuses are as follows: 0: (Unknown) 1: (Unused) 2: (Ignored) A pending construction or decommissioned conductor. 3: (Energized) A conductor that is energized. 4: (De-Energized) A de-energized conductor. 5: (Suspect Open) and area that is suspected to be deenergized, but the upstream device has not been confirmed to be open. 6: (Parallel) A conductor fed by a set of parallel feeders. 7: (Bidirectional) A conductor fed from both sides, either due to a loop or the direct path of a parallel. 8: (Meshed) A conductor fed by at least one feeder marked with an MID. 9: (Degraded) A partially energized conductor. 10: (Phase A) Phase A is hot. Used in Phase Coloring mode only. 11: (Phase B) Phase B is hot. Used in Phase Coloring mode only. 12: (Phase C) Phase C is hot. Used in Phase Coloring mode only. 13: (Phase AB) Phases AB are hot. Used in Phase Coloring mode only. 14: (Phase AC) Phases AC are hot. Used in Phase Coloring mode only. 15: (Phase BC) Phases BC are hot. Used in Phase Coloring mode only. 16: (Multistate) A combination of Grounded, Faulted, Parallel, etc. on separate phases. 17: (Grounded) The conductor is grounded. 18: (Faulted) The conductor is grounded and energized, or fed by two different phases connected together. 19: (Trace) The conductor is part of the trace results requested by the user. 20: (Isolated) The conductor is in an isolated segment. Isolated segments are areas that are bounded by devices that have been designated as isolated via a Control Tool action and, optionally, any devices that inherit from the "implicit_isolate" base class. 21: (Delegated) The conductor is in a delegated zone. Delegated Zone areas are bounded by devices delegated from a safety document and, optionally, any devices that inherit from the "implicit_delegate" base class. 22: (Secure) The conductor is in an secured segment. Secured segments are areas that are bounded by devices secured from the Control Tool and, optionally, any devices that inherit from the "implicit_secure" base class. 23: (Abnormal Radial) The conductor is fed by only one MID feeder.
off_nominal_flag	0/1: Whether the specified state/symbology_id is for an off-nominal position.

Column	Description
symbology_id	The symbol ID to use. This is either the conductor coloring identifier or the symbol ID found in the symbol file.

Note: All non-conductor/non-switch objects must use state "0" and off_nominal_flag "0".

The QUALITY_RULES Table

The QUALITY_RULES database table contains the mapping from device class and quality code to the desired symbol or text.

Column	Description
priority	The priority of the symbol. If more than one symbol matches exactly, the lowest priority rule will be used.
value	The integer value of the quality.
string	The string to display, usually a single character.
description	A description of the quality.
color	The color of the text to use.
location	The location (1-9) to use.
symbol	The symbol ID to use, or -1, if text is to be used; use 0 if no text or symbol should be displayed.
off_nominal	Unused

The CONDITION_RULES Table

The CONDITION_RULES database table contains the mapping from device class and condition type to the desired symbol.

Column	Description
condition	The condition class name, or 'digital' for digital SCADA measurements, 'analog' for analog_rules.
class	The device class.
devsymbol_num	The analog or digital attribute number, used only for measurements; 0 otherwise.
priority	The priority of the symbol. If more than one symbol matches exactly, the lowest priority rule will be used.
location	The location (1-9) to use. These correspond to upper-left, upper, upper-right, left, center, right, lower-left, lower, lower-right.
status	The condition status or digital value; -1 for analogs. For truck_location conditions, this is the (TYPE_NUM-1) value from the CREW_TYPES table.
symbol_num	The symbol to use. Only valid for conditions and digitals; -1 for analogs.

DMS Data Model Requirements and Configuration

This section describes the DMS Data Model process and data requirements. The data discussed in this section will be used by all DMS applications: Power Flow, Suggested Switching, Network Optimization, Feeder Load Management, Fault Location Analysis, and Fault Location Isolation and Service Restoration. It is intended to assist the customer during the DMS data modeling process. It is meant as an introduction, rather than a comprehensive reference.

Data Import and Build Process

The Oracle NMS shares a common model between the OMS and DMS applications. When a customer decides to implement DMS functionality they will need to layer engineering attributes (for example, impedances, voltage, ratings, and so on) onto the existing OMS connectivity model such that a power flow analysis can be conducted. The engineering attributes can be provided from GIS data, planning system data, protection coordination data, or a combination of multiple data sources. For data that is not readily available the Oracle Power Flow Engineering Data workbook can be used to supplement what may be missing.

The power flow runtime table configuration is created using the normal NMS model configuration practices and the data is populated during the normal build process. Initially the implementer needs to determine the availability of engineering attributes in the GIS and supplement what may be missing using flat files from an ancillary source (*e.g.*, planning system) or provide the missing data in the Power Flow Engineering Data workbook. The DMS model configuration defines logic that determines from what source data should be obtained. For example the build process should first look in the GIS data for transformer kVA ratings, if this is determined to be null the process should next look in the Power Flow Engineering Data workbook tables for a device specific value, if this is determined to be null use a default defined in the workbook table. The logic is custom configured by the implementer for each attribute for each device type. The data requirements for each device type are discussed in the Modeling Device Data section below.

kVA Solutions

The data requirements for full power flow functionality are extensive and can take a significant amount of effort to validate before being cleared for use in production systems. A simpler data model that allows NMS to perform only kVA solutions (using nominal voltage) is also available.

kVA mode for solutions is configurable on a per-feeder basis through the **Feeder Management** tab of the Configuration Assistant. This allows the power flow engine to be rolled out into areas of the model as and when the data becomes available.

The following applications are supported using a kVA based solution: Power Flow, Suggested Switching, Feeder Load Management, and Fault Location Isolation and Service Restoration.

> **Note:** Optimization and Fault Location Analysis will not function with a kVA based solution a full power flow model is needed for these two applications.

The following sections describe in detail the modeling and attribution required for all device classes for full power flow functionality. The reduced data set required for kVA solution is also discussed. Attributes that are not required for kVA solutions should be defaulted to reasonable values to enable consistency and validation checks to pass.

Modeling Device Data

This section will discuss each device type and the attributes required. The implementer will need to determine where this data can be obtained and think about the logic that will used in the model configuration for each attribute.

Sources

The customer must provide an equivalent source model for each source node defined in the data model. These source nodes represent constant voltage buses that are used to determine energization of the system and are generally located at feeder heads, the substation secondary bus generation sources, or the substation primary side bus. The required equivalent source parameters must represent an equivalent impedance looking up into the transmission system from the source node in question, in addition to voltage magnitude and angle. It is possible to model equivalent sources as zero impedance (*i.e.*, 'infinite bus'), but this will impact the accuracy of short circuit calculations provided by the Power Flow solution. kVA modeling requirements for sources are: nominal voltage and initial angle.

Line Impedances

The customer must provide line data in one of three ways within the Power Flow Engineering Data workbook.

- The first and preferred way is to provide the phase impedance data for each three-phase line type. The phase impedance data must be the self and mutual impedance and shunt susceptance for each phase. The phase impedance data must be provided in the Line Phase Impedance tab of the power flow engineering data workbook. Oracle Utilities Network Management System Power Flow Extensions supports the modeling of symmetric and asymmetric lines. Lines are considered symmetric when the 3 phases have the same conductor type. Lines are considered asymmetric when the 3 phases have at least two different conductor types.
- The second way is to provide the sequence impedance data for each line. The sequence impedance table data must be the positive and zero sequence impedances and shunt susceptance for each line. The sequence impedance data must be provided in the Line Seq Impedance tab of the power flow engineering data workbook.
- The third way only applies to overhead lines and is used to provide the conductor and construction types for each line; this method **cannot** be used for underground cables. The power flow engineering data workbook uses Carson's Modified Equations to calculate phase impedance data from these inputs. This data must be provided in the Line Conductors tab and Line Constr Impedances tab of the power flow engineering data workbook. Line Conductors contains the individual conductor characteristics and limits associated to a conductor type. The Line Constr Impedance tab contains information regarding spacing and the types of conductor spacing combinations that occur in a particular customer data model.

For kVA solutions, all line impedance data can be defaulted to a generic conductor type. Line impedances are not used in the kVA engine.

> **Note:** Line lengths will be automatically calculated by the NMS model build process and placed in the `LENGTH` column in the `NETWORK_COMPONENTS` table. If a project wishes to override this value, they can set up an override table (`PF_LINES`), which provides a mechanism to override the calculated value with a project defined value (for example, GIS attribute). If a value is populated in the `PF_LINES` table, the analysis will chose that value over the calculated value in the `NETWORK_COMPONENTS` table.

Transformers and Regulators

Oracle Utilities Network Management System Power Flow Extensions supports explicit modeling of multiple forms of power transformers, such as auto transformers, load-tap-changing transformers, step-up/step-down transformers and regulators. Each of these types of transformers and regulators require transformer characteristic data provided by the customer. The data should come from a combination of GIS data and/or PF Workbook data utilizing `pf_xfmr_types` and `pf_ltc_xfmr` tables. This data is written to `pf_xfmr`, `pf_xfmr_tanks`, and `pf_xfmr_limits` tables by the model preprocessor, which are read directly by Power Flow Service.

For kVA solutions, the data requirements for Transformers and Regulators are: Primary voltage, Secondary voltage, and rating.

Distributed Generation

Oracle Utilities Network Management System supports modeling of distributed energy resources for use in the power flow analysis. The runtime data for distributed energy resources is stored in table `pf_dist_gen` which is read directly by Power Flow Service. The parameters used within this table can be provided either via the customer's data source(s), typically the GIS or via the Power Flow Workbook using table `pf_dist_gen_data`. The configuration supports small scale residential level generation all the way up to large spinning units that may be located at the substation. Generators must be configured with generation curves which represent the generator output for a particular condition. For example, a customer may configure three generation curves to categorize wind generation for Windy, Partly Windy, and No Wind. Using the Weather Zone Forecast Tool, a customer can set the gen schedule for the next 7 days, which drives what profiles are used for each type of generation.

For kVA solutions, the data requirements for Distributed Generation devices are: Rating and zone.

Capacitors and Reactors

Shunt (capacitor/reactor) parameters must be provided from the customer's data source(s), typically the GIS. These parameters are defined in the `pf_capacitor_data` table, which will be accessed by the Model Preprocessor, which in turn will write the `pf_capacitors` table, which will be read directly by the Power Flow Service.

Oracle supports the following types of shunt regulation:

- **Voltage switched shunts:** local or remote bus regulation
- **Current switched shunts:** local line or cable regulation
- **kVAr switched shunts:** local line or cable regulation
- **Power-factor switched shunts:** local line or cable regulation
- **Temperature switched shunts:** on and off temperature
- **Time of day switched shunts:** on and off time of day.
- **Fixed shunts:** no regulation
- Shunt sequence control

The supplied data for each shunt must indicate which type of regulation is to be used and the corresponding control attributes.

For kVA solutions, Capacitors and Reactors essentially need to be modeled in full.

Switch Ratings

Switch rating parameters must be provided from the customer's data source(s), which is typically a GIS. These parameters are defined in the `pf_switch_data` table, and are accessed by the Model Preprocessor, which writes to the `pf_switches_table`, which is read by the Power Flow Service.

Predicting and reporting violations is a core advantage of the NMS distribution management system. Having accurate ratings and protection settings will allow the applications to generate meaningful violations. Data can be defaulted to reasonable values. Alternatively, they can be modeled as "Zero Impedance" devices to stop violation checking.

It should be noted that for protective devices that open to clear faults such as breakers, reclosers, and fuses the minimum pickup current should be used. For example, a breaker may be rated for 2000Amps, but the relay could be configured to trip at values greater than 600Amps; in this situation, a value of 600Amps should be configured for the devices normal, emergency short term, and emergency long term ratings.

For non-protective devices, the max continuous current rating of the device should be used which would represent the maximum amount of current the device could handle before failing. If desired, a separate emergency short term and long term limit could also be provided for non-protective devices. These values would represent a limit that could be sustained for a short duration of time (for example, 2-4 hours for short term and 24 hours for long term).

Within NMS, the capability exists to dynamically determine if protection reach issues exist on feeders. This occurs when the fault current on portions of a feeder are lower than the minimum pickup settings configured for the relay. In this situation, the fault current would not be seen as fault current and the device would not open to clear the fault. This can occur when feeders are switched abnormally and the length extended by picking up portions of adjacent feeders. To take advantage of this functionality, customers will need to configure the minimum pickup current for ground faults and for phase to phase faults for protective devices. For non-protective devices, or if this is not desired functionality, the values can be left null

Modeling Loads

Load modeling consists of basic load data which is used to determine average loading levels and load profile data, which is used to provide detailed information for load variation over time.

Basic Load Data

During modeling efforts, loads must be assigned to specific equipment types. The preferred approach is to insert a load (supply node) at the secondary of each underground and overhead distribution transformer. Supply nodes may also be created at primary metering points for cases where there is no voltage transformation or transformation is unknown or customer-owned.

For each load, a utilization factor can be specified, which represents the average loading level for the rated size of the transformer. For the most accurate power flow results, this data should be based on per-instance consumption data, which can often be obtained from historical billing information by dividing the total energy consumption of attached customers by the billing period and transformer rating.

The power flow data for load is defined using the table pf_load_data, which is read by the Model Preprocessor, which in turn writes the pf_loads table, which is read by the Power Flow Service.

Load Profile Data

Load profile data is used to model how load changes over time. A single load profile represents the change in load levels over a 24-hour period. Multiple profiles may be associated with a single load to represent different load behavior for different types of day (*e.g.*, weekday, weekend) and for different temperatures (*e.g.*, 80°F - 90°F, 90°F - 100°F). The use of load profile data improves the accuracy of the DMS applications by providing more realistic loading scenarios for the current or predicted analysis time period. For example, profiles are used to verify switch plans, determine suggested switching recommendations, and generate daily and seasonal peak limit alarms.

The Oracle Utilities Network Management System supports a variety of sources of load profile data such as load class profiles or individual transformer profiles. Once processed, all profile data is placed in the pf_load_interval_data table. For load class profiles, the profile_id column in the pf_loads table can be set to point to the appropriate data in the pf_load_interval_data table. For transformer specific profiles, the adapter will populate a table pf_loads_profile_override; this table will be used to specify which transformers have specific profiles as opposed to just load class profiles. Anything defined in the pf_loads_profile_override table will override the profile_id field in the pf_loads table.

Load Class Profiles

Load class profiles represent typical load changes over time for a particular type or class of load, such as residential, commercial and industrial. This type of profile data can be obtained from general sources, or the customer can collect this data from typical customers or feeders. When using load class profiles, the load level at each load point is determined by combining the rated kVA with the load utilization factor and the class profile associated with that load. Load class profiles are useful where detailed data for each load is unavailable.

Transformer Profiles

Modern Meter Data Management (MDM) systems make it possible to collect detailed power usage histories for each customer. By aggregating individual meter loads to each service transformer, it is possible to create detailed load profiles for each transformer location. This data can be derived from either representative historical conditions or using predictive values, if the MDM system has this capability.

When using transformer profiles, all load data is derived from these profiles and basic load data such as the utilization factor is not used. The load profile input data can include both kW and kVAr values for load over the 24 hour period.

For kVA solutions, load modeling is still necessary. The amount of load modeling required will depend on the applications being used. If reasonably accurate future forecasting is required for Feeder Load Management, FLISR and/or Suggested Switching then detailed load profile curves will have to be generated. For real-time load flows only rating, utilization factor and power factor are required. Load will be scaled from SCADA data (where available).

Temperature Based Profiles

For all load profile sources, a temperature parameter can be used to add more granularity to the source data. Multiple profiles can exist that categorize how load behaves for a particular temperature range (*e.g.*, 80°F-90°F) for each day type. The power flow solution will use a zone based temperature forecast to determine what profile to use for each hour. For example, if it is forecast to be 60°F on a Monday at 9 a.m., the power flow solution would use the profile factor from the profile for a 60-70°F weekday morning at 9. For each day, multiple pieces of load profiles will be used to create a composite profile that represents the temperature profile for that day. Each load needs to be tied to an associated weather zone and the granularity at which temperature bands are defined will need to be defined in the pf_temperature_bands database table (*e.g.*,10°F, 20°F, 5°F).

NMS provides a productized load profile adapter that assists with the creation of temperature based profiles. The adapter maintains a history of previously recorded load profile data and is continuously adjusting and updating profiles as new data is received. It adaptively learns what profiles look like for different day type and temperature combinations. The more data that is gathered by the adapter, the more accurate the profiles become. To use the adapter, you need to provide raw profile data aggregated to the transformer/primary meter level in a CSV file that matches the file format specified by Oracle (see the *Oracle Utilities Network Management System Adapters Guide* for more information). The adapter will determine what day type and temperature the profiles map to and then merge the data with existing data in the database. For example, if a new sample is received for Monday at 9 a.m. for profile "Sample_xfmr_1" at 65°F, the adapter needs to determine if the sample already exists; if it exists, the adapter calculates a rolling average based on the total number of samples received. Modeling Distributed Generation

Distributed Generation modeling consists of configuring parameters related to the unit that characterizes its power output. These parameters are defined in the *pf_dist_gen_data* table, which will be accessed by the Model Preprocessor, which in turn will create the *pf_dist_gen* table, which is read directly by Power Flow Service. If desired a utility can setup the model preprocessing such that all or only some of the attributes provided come from the GIS rather than the *pf_dist_gen_data* table. Distributed generation resources need be configured with the proper rating, impedance, power source (*e.g.*, Wind, PV, Fossil Fuel, etc), and voltage regulation data that characterizes the unit. Each DER is also mapped to a set of profile data, which is used to provide detailed information regarding how a DER generates power over time. In certain situations such as a battery the profile will show when the unit is charging (consuming power) versus generating power.

Distributed Generation Profiles

Distributed Generation profile data is used to model how much power is contributed or even consumed over a period of time. Multiple profiles can be associated to a distributed generation resource to characterize how it behaves during different situations. For example a PV dist gen may have 6 different profiles to represent various levels of cloud cover (*e.g.*, Clear, Partly Cloudy, Cloudy, etc). Each DER will also be associated to a particular power source to represent the type of distributed generation, for example Wind and PV may be project configured power sources. When forecast data is changed the power source attribute allows the Power Flow Service to determine what DERs need to have a profile change. For example if an admin changes the current days forecast from clear to cloudy you would only want to change the profiles for the PV dist gens but not the wind dist gens since that power source is unaffected as the wind speed is to remain the same. The use of distributed generation profiles improves the accuracy of the DMS applications by providing more realistic values of how distributed generation may actually be behaving in the field. For example PV output will be less during the early morning or late afternoon hours and nonexistent at night. During a cloudy day the output will be less than on a clear day. All of these situations can be handled with properly configured profiles.

Generation of PV Profiles

If desired a utility can use the product profile adapter to create PV profiles based for a predetermined set of day types (e.g. Cloudy, Clear, Partly Cloudy, etc). When PV profile creation is configured the adapter will create PV profiles based on a set or specified configuration options. The PV profile creation will be based on a specified lat/long and time of year, this way each time the adapter is run a new set of accurate PV profile curves will be generated and loaded into the DB for use by PFService. A command line option (-createPvOnly) when running the adapter will also exist to force the adapter to just generate the PV profiles and not load any load profiles into the DB.

Net Metered Profiles

If utilities are providing net metered profiles the Profile adapter can be configured to add back in the load served by the residential distributed generation resources such that a true load profile is provided to NMS for use within the power flow analysis. For example if you have a service transformer serving one customer and that customer consumes 100kW. If PV is present at that customer premise producing 40kW on a sunny day the power required from the utility to meet the load is 60kW. In this case the meter only telemeters 60kW even though the customer consumed 100kW. Since loads need to be split from distributed generation resources in the NMS network model we want the true load consumed by the customer not the net metered load. The adapter will be responsible for identifying dist gens below a service transformer and add the load served back into the net metered profile.

Catalog Tables

The catalog tables identified in this section must all be populated by the customer. The Power Flow Engineering Data Excel workbook should be used as a template to assist the customer in identifying source data locations (planning power flow data, database tables and so on), defining a data export mechanism, and specifying the Oracle table names, columns, and data formats into which the source data must be imported. See the example workbook in the Oracle Utilities Network Management System product directory location: $NMS_BASE/OPAL/workbooks.

DER and the Weather Zone Forecast

A utility has the ability to provide forecast data for various types of DER resources that are present within the electrical network. The NMS categorizes DERs into the following forecasts categories:

- Weather Affected DERs
- Utility Scale DERs
- Demand Response Groups

The following sections will describe how to configure them, and provide the required forecast data.

Weather Zones

A utility has the ability to configure different weather zones within the NMS data model such that different distributed generation profiles can be used within each of the different zones to characterize the real-time and forecasted generation output. For example, if the "coastal" zone is cloudy, a user could set the forecast for PV to use a cloudy profile in the coastal zone, but all other zones would use the clear profile. The weather zone configuration is also used by loads to adjust load profiles as temperature changes. DERs technology types (*e.g.*, PV, Wind, *etc.*) that are mapped to weather zones should be weather affected DERs only. For example, a large utility scale diesel generator would not be mapped to a weather zone since its forecast and output is independent of weather conditions.

If a utility wishes to configure weather zone functionality, data needs to exist that maps each distributed generation and load resource in the NMS data model to one of the applicable zones. The preferable location for this data would be the GIS, but, if needed, the data could be placed in the customer specific Power Flow Engineering Data workbook. This data needs to be brought across during the model build into the ZONE columns of the PF_DIST_GEN and PF_LOADS tables. The value populated will need to map to a configured zone in the PF_WEATHER_ZONE table.

Once Weather Zones have been configured, the utility has the ability to provide a forecast for each zone that forecasts how the DER resources for each power source will behave for the current day plus the next six days. This forecast should also include a temperature forecast for each applicable zone; this will be used to adjust load profiles as temperature changes.

During implementation, a utility needs to determine how they would like to use the weather zone forecast functionality.

- They could have an administrator use the Weather Zone Forecast tool to set the generation profile daily for each zone and power source. The power flow solutions will then use the profiles that have been set for the day in the real-time solutions and forecasts.
- For more advanced forecasting, they could use an external interface (*e.g.*, weather feed) to populate the PF_WEATHER_ZONE_FORECAST table for each hour of the day with what generation profile to use. This would allow for more granular refinement of the power flow solutions. In this scenario the utility would not use the Weather Zone Forecast tool to set the weather forecasts.
- Alternatively, they could provide direct scaling factors to use for each hour instead of what profile to use for each hour; the input method can be configured with the distGenDefault SRS rule.

Whichever method is used, the entire forecast for each zone must be provided with the same method (profiles or scaling factor) for each power source.

Temperature forecasts can be provided for US customers using a product adapter that interfaces to a weather data server. Alternatively, an implementer could build a custom adapter that would provide temperature forecasts on a zone basis from an alternate resource such as utility specific forecasts (see the *Oracle Utilities Network Management System Adapters Guide* for more information).

Large Scale Utility DERs

Large scale utility DERs are categorized as units that are large in scale and unaffected by weather. An example would be a diesel generator or large battery that has an output that is unaffected by weather. For these units, the forecast is provided for each individual unit and they are not aggregated together like DERs within weather zones. It may be entirely plausible to have two large batteries in close proximity to each other but their forecasts could be drastically different.

If a utility wishes to categorize a DER resource as being outside of a weather zone the ZONE field within the `PF_DIST_GEN` database table should be configured within the alias of the unit. The preferable location for this data would be the GIS, but if needed the data could be entered in the utility's Power Flow Engineering Data workbook. This data will need to be brought across during the model build. A corresponding power source also needs to be configured for the unit (for example, Diesel, Gas, Battery, etc.)

Once the large scale DERs have been configured, the utility has the ability to provide a forecast for each unit that forecasts the output for the current day plus the next six days. It is expected that an external system would be used to populate the forecasts for these units and the data is stored in the `PF_DER_FORECAST` database table. Within this table a key corresponding to a distributed generation profile can be provided (for example, on, off, peak shave, and so forth). Alternatively, a utility could provide direct scaling factors to use for each hour instead of what profile to use for each hour, the input method can be configured with the distGenDefault SRS rule. With either method the entire forecast for all units must be provided with the same method (profiles or scaling factor) for each power source. The NMS has a DERMS adapter product that is capable of taking a CSV or RDBMS based forecast from an external system for use by the NMS (see the *Oracle Utilities Network Management System Adapters Guide* for more information).

Demand Response Groups

A utility has the ability to configure different demand response groups (DR groups) within the NMS data model such that different DR profiles can be used within each of the different groups to characterize the real-time and forecasted DR scenarios. For example, the ability to simulate a DR group dropping load for 2 hours during a peak load day.

If a utility wishes to configure demand response group functionality, data would need to exist that maps each DR resource in the NMS data model to one of the applicable demand response groups. The preferable location for this data would be the GIS, but, if needed, the data could be placed the utility's Power Flow Engineering Data workbook. This data will need to be brought across during the model build into the ZONE column of the `PF_DIST_GEN` table. The value populated will need to map to a configured zone in the `PF_DEMAND_RESPONSE_GROUPS` table.

Once the demand response groups have been configured, the utility has the ability to provide a forecast for each demand response group that forecasts the output for the current day plus the next six days. It is expected that an external system would be used to populate the forecasts for these units and the data is stored with NMS DB table `PF_DEMAND_RESPONSE_FORECAST`. This table can have a key corresponding to a distributed generation profile (for example, on, off, etc.). Alternatively, a utility could provide direct scaling factors to use for each hour instead of what profile to use for each hour; the input method can be configured with the distGenDefault SRS rule. With either method, the entire forecast for all units must be provided with the same method (profiles or scaling factor) for each power source. The NMS has a DERMS adapter product that is capable of taking a CSV or RDBMS based forecast from an external system for use by the NMS (see the *Oracle Utilities Network Management System Adapters Guide* for more information).

DER and Weather Forecast Tables

PF_WEATHER_ZONE

Column	Comments
ZONE_NAME	Name of the generation zone (e.g. Desert, Coastal, Mountain)
DISPLAY	Boolean to indicate whether or not to display the zone on the user interface. If set to 'Y' display the zone if 'N' do not display the zone. This would be used by a utility that has many zones with some of them being fed by an automated weather feeder and others that are not. With this the utility could filter some of the zones out they don't need to display.
EDITABLE	Boolean to indicate whether or not the zone is user editable using the user interface. If set to 'Y' allow a user to edit generation profiles for the zone from the user interface. If 'N' do not allow a user to edit the generation profiles for the zone, this would generally imply the zone is being set from an automated weather feed. This would be used by utilities that have many zones with some of them being fed by an automated weather feeder and others that are not. With this the utility could filter some of the zones out they don't need to edit.
Latitude	The latitude of the center of the weather zone.
Longitude	The longitude of the center of the weather zone.

PF_WEATHER_ZONE_FORECAST

Column	Comments
ZONE	Zone the forecast is applicable to.
DATE_TIME	The date and time the forecast is applicable to.
POWER_SOURCE_X (depends on the number of power sources configured such as wind, PV, etc.)	Key to the profile for the specified fuel type. Will be an integer value that represents a profile configured for the power source. Alternatively if the power source is configured for scale factors as the input this will be a float representing the scaling factor.
TEMPERATURE	The temperature in either Fahrenheit or Celsius for the day, zone, and hour. This is used to adjust load profiles.

PF_DEMAND_RESPONSE_GROUPS

Column	Comments
DR_GROUP_NAME	The name of the demand response group.
DISPLAY	Boolean to indicate whether or not to display the zone on the user interface. • Y: display the zone. • N: do not display the zone.
MW_LOAD_DROP	The amount of load in MW that can be dropped with the group.

PF_DEMAND_RESPONSE_FORECAST

Column	Comments
DR_GROUP_NAME	Group the forecast is applicable to.
TEMPERATURE	The temperature forecast for the demand response group.
DATE_TIME	The date and time the forecast is applicable to.
POWER_SOURCE_TYPE	Key to the demand response type.
FORECAST_VALUE	Will be an integer value that represents a profile configured for the power source. Alternatively, if the power source is configured for scale factors as the input, this will be a float representing the scaling factor.

PF_DER_FORECAST

Column	Comments
RESOURCE_NAME	The name of the resource the forecast is applicable to.
DATE_TIME	The date and time the forecast is applicable to.
POWER_SOURCE_TYPE	Key to the generator power source type.
FORECAST_VALUE	Will be an integer value that represents a profile configured for the power source. Alternatively, if the power source is configured for scale factors as the input, this will be a float representing the scaling factor.

Temperature and Seasonal Limits

The power flow analysis can be configured to work with seasonal or temperature based limits when specifying ratings for power transformers and conductors. For these types of devices the upper limit at which the device can be pushed to is heavily dependent upon the ambient air temperature, in general these devices can be pushed harder when the temperature is cooler since natural cooling can occur. Both the seasonal and temperature based approach represent a mechanism to specify what the ambient air temperature is doing with the temperature based approach providing much more possibility for granularity. Both approaches are global in nature and a project cannot specify one type of configuration for one area of the model but another approach for another area. Any ratings provided for switches will just have the standard concept of normal, short term emergency, and long term emergency limits.

Configuration of Seasonal Limits

Seasonal Limits define a set of ratings that are applicable for predefined seasons. The power flow analysis will utilize the proper ratings based on the date of the power flow solution. For example, a project may configure two seasons (Winter and Summer); during the winter months, the defined winter ratings will be used for transformers and conductors, and for the summer months, the summer ratings will be used. Seasonal Limits can be enabled in the power flow analysis by following the steps below.

1. The USE_TEMP_LIMITS SRS rule should be configured with a value of "**no**" to enable seasonal limits within the analysis and also enable any user interface items that allow the user to specify the season.
2. The PF_SEASONS_DEF table needs to be configured and is used to store information related to season definitions used by power flow. The main points of configuration will be the season name and the start and end dates. The PF_SEASONS_DEF table should be defined in the `[project]_pf_seasons.sql` file.

PF_SEASONS_DEF

Column	Comments
SEASON_NUMBER	The integer key corresponding to the season
SEASON_NAME	The name of the season (for example, Summer, Winter)
SEASON_START_DAY	The integer start day of the season
SEASON_START_MONTH	The integer start month of the season
SEASON_END_DAY	The integer end day of the season
SEASON_END_MONTH	The integer end month of the season

3. The integer value assigned to each season will need to correspond back to the same integer key assigned to transformer and conductor seasonal limits defined in the Power Flow Engineering Data workbook. The relevant transformer limits will be defined on the workbook's **Power Xfmr Limits** tab. The relevant conductor limits will be defined on the **Line Phase Impedances**, **Line Seq Impedances**, and/or **Overhead Line Conductors** tabs, depending on which method was used to provide the conductor impedances data. For each season, a set of limits will need to be provided for a *normal*, *emergency short term*, and *emergency long term* rating.

Configuration of Temperature Limits

Temperature Limits define a set of ratings that are applicable for predefined temperature bands. The power flow analysis will utilize the proper ratings based on the forecast temperature for the date and time of the power flow solution. So as an example the configuration may be setup such that conductors and transformers will have a limit defined for each 20°F range of temperature. The limit used for each power flow solution will be based on the forecast temperature for the date and time the solution is for and weather zone the device is in. Therefore while looking at forecast power flow solutions for a device is may be possible to see the upper limit changing as the temperature changes throughout the day. This is the one big differentiator between seasonal based limits and temperature based limits, with the temperature based limits the limit used has the possibility to change hour by hour as temperature changes whereas seasonal limit the same limit will be used for a larger date range. Temperature Limits can be enabled in the power flow analysis by following the below steps.

1. The USE_TEMP_LIMITS SRS rule should be configured with a value of "**`yes`**" to enable temperature limits within the analysis and also enable any user interface items that allow the user to specify the temperature.
2. The PF_TEMPERATURE_BANDS table will need to be configured and is used to store information related to the number and range of each temperature band. The main points of this table will be setting an integer value that represents the band and the upper limit of the band; the lower limit will be based on whatever is configured as the upper limit from the previous band. The units on this table do not matter as long as the units when providing the data are all kept consistent; for example, when defining temperature bands, if the units could be in Fahrenheit or Celsius as long as the weather feed providing the forecast data is providing data with the same scale, the proper limit will be chosen.

   Note: This table is shared configuration with defining of temperature bands for load profiles so the IS_LIMIT column needs to be set to "**`Y`**".

PF_TEMPERATURE_BANDS

Column	Comments
TEMP_INDEX	The integer key corresponding to the temperature band
UPPER_LIMIT	The upper limit of the temperature band
IS_LIMIT	Specifies Y/N if this temperature band is for limits or profiles

3. The integer value assigned to each temperature band will need to correspond back to the same integer key assigned to transformer and conductor temperature limits defined in the Power Flow Engineering Data workbook. The relevant transformer limits will be defined on the workbook's **Power Xfmr Limits** tab. The relevant conductor limits will be defined on the **Line Phase Impedances**, **Line Seq Impedances**, and/or **Overhead Line Conductors** tabs depending on which method was used to provide the conductor impedances data. For each season, a set of limits need to be provided for aa *normal*, *emergency short term*, and *emergency long term* rating.
4. A weather feed or custom integration will be needed to populate PF_WEATHER_ZONE table with temperature forecast information for each weather zone. This weather feed will be used by the power flow analysis to determine which limit to use for the real-time power flow solutions and forecast solutions; the forecast temperature will drive which of the predefined limits is used. For example, if a power flow solution being conducted

at 9:30 a.m. has a forecast temperature of 65°F, the predefined rating for the 60°F-70°F band will be used.

Note: It is possible to configure a temperature measurement for SCADA telemetered power transformers. In this situation, the analysis will use the SCADA-provided temperature rather than the forecasted temperature for the device's weather zone when choosing the limit in the real-time solution. For all forecast solutions, the weather feed temperature will still be used for choosing the limit.

5. Power transformers and conductors need to be mapped to a relevant weather zone such that the analysis can determine the applicable temperature to use for each device. The weather zone for transformers is mapped to the ZONE column in the PF_XFMRS table. The weather zone for lines is mapped to the ZONE column in the PF_LINES table.

Bellwether Meter Voltages

In real-time mode, Optimization is capable of making use of bellwether meter voltages. The voltages will be used as an input into the optimization algorithm to ensure that voltage violations are not created on the secondary network serving customer meters. By using voltage telemetry, the Oracle Utilities Network Management System can begin to learn the difference between the solved power flow voltage versus the actual telemetered voltage and make estimates as to what the secondary service drop may be between a service transformer and the bellwether meter location. If a customer strategically choses bellwether meter locations at points that experience the lowest voltage on a feeder, it will help ensure that the optimization does not create voltage issues in scenarios like Conservation Voltage Reduction (CVR) where the voltage is lowered to reduce demand. Oracle recommends that customers choose two to three metering points per regulation zone on a feeder. These voltages can come from field devices (for example, sectionalizers, reclosers, capacitors, and so forth) that are capable of telemetering voltage, but would be most useful at customer meters. For more information on configuration of the bellwether meter voltages, please see the **SCADA Configuration for Bellwether Meters** section in the **Building the System Data Model** chapter of the *Oracle Utilities Network Management System Adapters Guide*.

Viewing Power Flow Attributes

Power Flow attributes can be configured to display within the NMS Attribute Viewer. This would enable a DMS customer to see power flow attributes in addition to normal GIS attributes when using this tool. The Model Workbook is setup to programmatically determine relationships between the model tables and join the related tables together when generating attribute view. The tool tab **Generate Attribute View with Supporting Tables** check box enables this functionality.

To enable/disable the attributes that will display within the NMS attribute viewer, a project needs to populate a display name in the *Display Name* column of each applicable power flow tab within the model workbook (*e.g.*, pf_loads, pf_dist_gen, pf_xfmrs, etc.).

It is recommended that projects use a key prefix in each of the display names so a user can easily see which attributes are GIS display-only and which are power flow attributes that are used in the power flow analysis. For example, a project could use a prefix of "PF_" on each power flow related attribute.

Power Flow Service High Level Messages and Command Line Options

Power Flow Service High Level Messages

High level messages are available for data maintenance and troubleshooting.

> **Note:** Oracle recommends that you check with Oracle Support before using these options for the first time - especially in a production environment. They could have unintended consequences for application functionality, performance, or environment resource consumption.

The following **Action** commands are accepted by **PFService**, using the form:

```
Action any.PFService <command> <options>
```

- **dump [<resource>]:** Dump various internal data stores, as determined by <resource>. If no resource is given, then the configuration rules are reported.
  - **measures:** Dump SCADA measurement information for all devices.
  - **memstats:** Dump memory statistics.
  - **nsm:** Dump NSM statistics.
  - **island <hdl>:** Dump contents of island containing <hdl>.
  - **feeder <hdl>:** Dump contents of island containing <hdl>.
- **feederstatus:** Dump feeder status.
- **reload:** Reload configuration rules.
- **reload_rules:** Reload configuration rules.
- **processintervaldata:** Reload Xfmr profile data.
- **process_dms_feeders:** Reload the DMS Feeder Status data.
- **validate_network:** Report on Powerflow Model data.
- **dump_model <device> [session]:** Topology dump of island containing device
- **island_report**:: Dump a status report about island solutions.
- **clear_scada:** Clear PFService of all SCADA updates.
- **reset_dgs:** Clear all temporary Distributed Generation conditions.
- **show_scada <device>:** Display SCADA measurements for a device.
- **np_emulate [on|off|0|1]:** Set Network Protector mode to on (1) or off (0). Mode toggles if no option given.
- **what <dev hdl | dev alias>:** Report PowerFlow and topology status of device.
- **study_what session <dev hdl>:** Report PowerFlow and topology status of device in study session.
- **debug [FACILITY} <level>:** Set debug reporting level for any of the following facilities:
  - **DBS:** Report database related activity.
  - **EMULATOR:** Emulator only based activity.
  - **FLA:** Report FLA related activity.

- **FLISR:** Report FLISR related activity.
- **FLM:** Report FLM related activity.
- **FLOW:** Report Nominal Flow Direction calculations.
- **LOCKS:** Report Lock/Mutex handling.
- **LOADPROFILE:** Report Xfmr profile handling.
- **MEASURES:** Report SCADA related activity.
- **MEMSTATS:** Report memory usage at significant events.
- **MESSAGES:** Report messages.
- **MODEL:** Report Model import, build and update actions.
- **PFSOLVE:** Report Power Flow solution activity.
- **SEGMENT:** Report island rebuilding actions.
- **SOLVE:** Report topology solution activity.
- **SS:** Report Suggested Switching activity.
- **VV:** Report Volt/VAr activity.
- **TIMING:** Report function timing.

  Note: If a facility is not specified, you can turn debug on for all of them using `Action any.PFService debug <level>`.

- **debug_device <device>:** Enable debug information for device.
- **stop_and_free [core]:** Free all memory, and stop [optionally produce core file].
- **processcatalogs:** Reload device rating catalogs.
- **resolve:** Resolve the current solution.
- **flisr <command>**
  - **dump:** Dump FLISR status information.
  - **alarms:** Perform the daily reminders about disabled devices.
  - **mode <newMode>:** Set the system wide mode for FLISR to one of the following:
    - **disable:** Disabled
    - **manual:** Manual
    - **automatic:** Automatic
- **device_disable <dev hdl> <user>:** Disable the device for FLISR processing.
- **device_enable: <dev hdl> <user>:** Re-enable the device for FLISR processing.
- **flm <command>**
  - **reforecast:** Re-execute all future FLM forecasts.
  - **hourlyTimer:** Perform the top of the hour tasks.
  - **stop:** Stop forecasting.
  - **start:** Restart forecasting.
- **feeder_ca <command>:** Control N-1 feeder contingency analysis.
  - **start <runtime>:** Start analysis, run for <runtime> minutes (default 60).
  - **stop:** Stop feeder contingency analysis.

- **feeder_da <command>:** Control feeder data analysis.
  - **start [runtime]:** Start data analysis, run for <runtime> minutes (default 60).
  - **stop:** Stop feeder data analysis.
- **purge_fca:** purge all of the existing Feeder Contingency Analysis results.
- **purge_fda:** purge all of the existing Feeder Data Analysis results.
- **solve_island <device>:** Solve island containing <device hdl>.
- **dump_disabled_islands:** Report list of islands disabled by FLM.
- **reenable_island <src hdl>:** Re-enable a specific island.
- **reenable_island:** Re-enable all non-convergent islands.
- **disable_island <src hdl>:** Disable a specific island.
- **messagestat <command>:** Control PFS message statistics.
  - **print:** Report statistics on PFService message usage.
  - **reset:** Reset message counters to zero.
- **vvo_current:** Initiate VVO current (fast cycle) execution.
- **vvo_hourly:** Initiate VVO hourly (slow cycle) execution.
- **vvo_update_plan <event hdl>:** Force an update on VVO plan for a passed event handle.

Power Flow Service Command Line Options

-no_alarms: Prevent generation of system alarms when FLM detects Feeder abnormal conditions.

-stats: Write Model Errors and Power Flow Island Statistics to the database.

-pfdbs: Use a dedicated DBService connection reserved just for PFService queries and updates in the RDBMS.

-incrSolveCutoff: Number of devices affected by a topology change that causes rebuilding to switch from incremental to full solves. Same behavior as the MTService option.

-stats: Have FLM generate solution and island statistics from the power flow solutions.

-allowAutoFlisrStartup: Allow FLISR to startup in automatic mode.

Spatially Enabling the Data Model for Outage Analytics

The ces_parameters table contains a set of attributes that are used to enable the NMS electrical model for Oracle Utilities Outage Analytics. The following table describes these attributes.

ces_parameters attribute	Description
MBS_GEO_SRID	The Oracle Spatial reference ID for the geographic spatial layer.
MBS_GEO_MINX MBS_GEO_MAXX MBS_GEO_MINY MBS_GEO_MAXY	The minimum and maximum values for the two coordinate systems.
MBS_LL_SRID	The Oracle Spatial reference ID for the lat/long spatial layer.
MBS_GEO_CSMAP_COORDSYS	The CS_MAP-defined geographic coordinate system.
MBS_LL_CSMAP_COORDSYS	The CS_MAP-defined lat/long coordinate system.
MBS_LL_MINX	Min X value of data, used for spatial indexing
MBS_LL_MINY	Min Y value of data, used for spatial indexing
MBS_LL_MAXX	Max X value of data, used for spatial indexing
MBS_LL_MAXY	Max Y value of data, used for spatial indexing
MBS_LL_TOL	Tolerance value, used for spatial indexing
MBS_LL_XTYPE	X Coordinate type (Typically X or LATITUDE)
MBS_LL_YTYPE	Y Coordinate type (Typically Y or LONGITUDE)
MBS_GEO_TOL	Tolerance value, used for spatial indexing
MBS_GEO_XTYPE	X Coordinate type (Typically X or LATITUDE)
MBS_GEO_YTYPE	Y Coordinate type (Typically Y or LONGITUDE)

NMS CIM Import and Export Tools

NMS has two tools to support import and exporting CIM data: cim2mp and CIMExporter, respectively.

CIM Import

The CIM import processor, **`cim2mp`**, feeds directly into the standard NMS model build process. It takes a CIM-formatted file and converts it into an NMS model preprocessor (mp) file. Once the files are in the .mp file format, the Model Engineer configures the rest of the model interface just as they would with any GIS-supplied `.mp` file.

Usage:

```
cim2mp [cim_file] [mpfile]
```

Example:

```
cim2mp 3513.rdf 3513.mp
```

CIM Export

The CIM export tool, **`CIMExporter`**, exports a specific set of components from the NMS .mb file in CIM/IEC .xml/.rdf file format. The resulting file should be able to be imported by a CIM-compliant model consumer.

Usage:

```
CIMExporter mbfile cim_file
```

Example:

```
CIMExporter 3513.mb 3513.rdf
```

The `CIMExporter` configuration file, `CIMExport.properties`, is located in the `$NMS_HOME/sql` directory. The product version of this properties file is installed by default. Copy the product file to your project/sql directory and make any changes needed. Run `nms-install-config` to install the project/sql version into the `$NMS_HOME/sql` directory.

> **Note**: running `nms-install-config` will overwrite the product version.

CIMExport.properties can be used to:

1. Map NMS classes to CIM classes.
2. Specify the NMS attributes to map to CIM attributes.
3. Enable catalog lookup for powerflow line values.

Preparing the NMS Model for Oracle Utilities Customer Self Service

The Oracle Utilities Customer Self Service application reads NMS materialized views to display NMS data. These materialized views are created in the `[project]-CSS-setup` script and refreshed using the `[project]-CSS-refresh` script. Note that the user_sdo_geom_metadata table needs to be updated for the new materialized views with the source table rows' *diminfo* and *srid* values.

Materialized Views

GEOGRAPHIC_OUTAGES

The GEOGRAPHIC_OUTAGES materialized view is created from the JOBS, DIAGRAM_OBJECTS, and the NETWORK_COMPONENTS tables.

Column Name	Data Type	Description
outage_type	VARCHAR	A description of the outage type. "Probable Service Outage", as mapped from the JOBS.status
num_customers_out	NUMBER	The number of customers out, as mapped from the JOBS.user_cust_out.
begin_time	DATE	The outage begin time, from JOBS.begin_time.
est_rest_time	DATE	The outage estimated restore time, from JOBS.est_rest_time.
last_update_time	DATE	The outage last updated time, from JOBS.last_update_time.
cause	VARCHAR	The outage cause, if available.
geometry	MDSYS.SDO_GEOMETRY	The geographic geometry, from DIAGRAM_OBJECTS.geo_geometry.

GEOGRAPHIC_OUTAGE_AREAS

The GEOGRAPHIC_OUTAGE_AREAS materialized view is created from the zip_codes, jobs, supply_node_log, and customer_sum tables.

Column Name	Data Type	Description
area	VARCHAR	The name of the outage area, from the CUSTOMER_SUM.zip_code, CUSTOMER_SUM.City_State, or CUSTOMER_SUM.User_Geographic_Loc.
area_type	VARCHAR	Area type choice; for example, zip code, county, city.
cust_served	NUMBER	The number of customers in the area, as summed from CUSTOMER_SUM.customer_count for that area.
cust_out	NUMBER	The number of customers out in the area, as summed from CUSTOMER_SUM.customer_count for the supply_nodes with outage,
num_outages	NUMBER	The number of distinct active JOBS table records.
earliest_begin_time	DATE	The earliest outage begin time in the area, from the JOBS.begin_time.
latest_est_rest_time	DATE	The last JOBS.est_rest_time for the area.
last_update_time	DATE	The last JOBS.last_update_time for the area.
geometry	MDSYS.SDO_GEOMETRY	The ZIP_CODES.geometry for the area.

GEOGRAPHIC_OUTAGE_STATUS

The materialized view GEOGRAPHIC_OUTAGE_STATUS is created from the JOBS table.

Column Name	Data Type	Description
report_date	DATE	The date this view was created – SYSDATE.
cust_served	NUMBER	The sum of all CUSTOMER_SUM.customer_count records.
cust_out	NUMBER	The sum of all active JOBS.user_cust_out records.
num_outages	NUMBER	The number of distinct active JOBS table records.

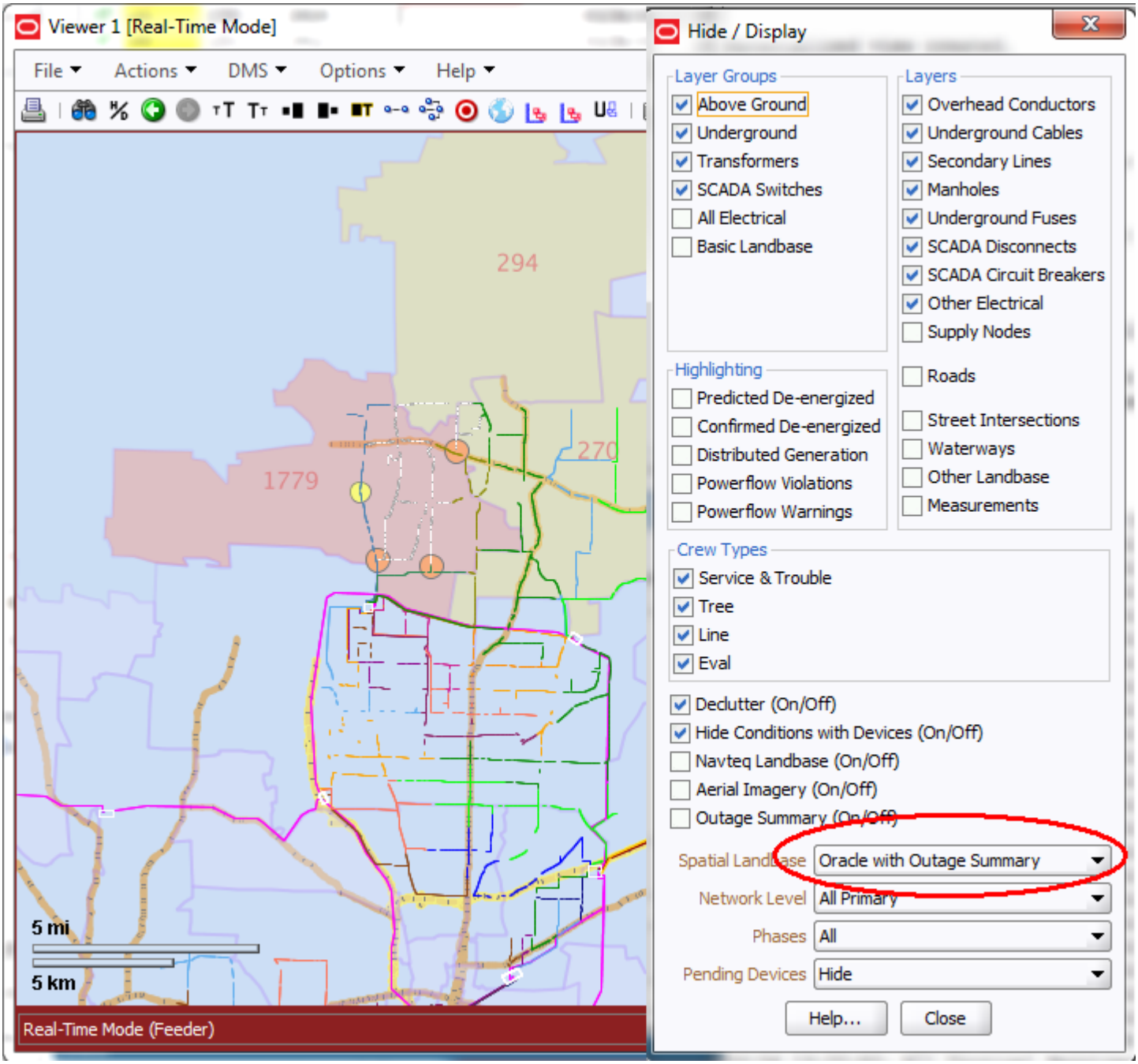
GEOGRAPHIC_OUTAGE_AREAS_D

The GEOGRAPHIC_OUTAGE_AREAS_D materialized view includes event details for each event in an area.

Column Name	Data Type	Description
area	VARCHAR	The name of the outage area, from the CUSTOMER_SUM.zip_code, CUSTOMER_SUM.City_State, or CUSTOMER_SUM.User_Geographic_Loc.
area_type	VARCHAR	Area type choice; for example, zip code, county, city.
event_id	NUMBER	The event id.
outage_type	VARCHAR	A description of the outage type. "Probable Service Outage", as mapped from the JOBS.status.
cust_out	NUMBER	The sum of all active JOBS.user_cust_out records.
begin_time	DATE	The outage begin time, from JOBS.begin_time.
est_rest_time	DATE	The outage estimated restore time, from JOBS.est_rest_time.
last_update_time	DATE	The last JOBS.last_update_time for the area.

Adding Outage Summary Spatial Landbase to the NMS Viewer

The Viewer has the ability to display Oracle Spatial data behind the Viewer maps:

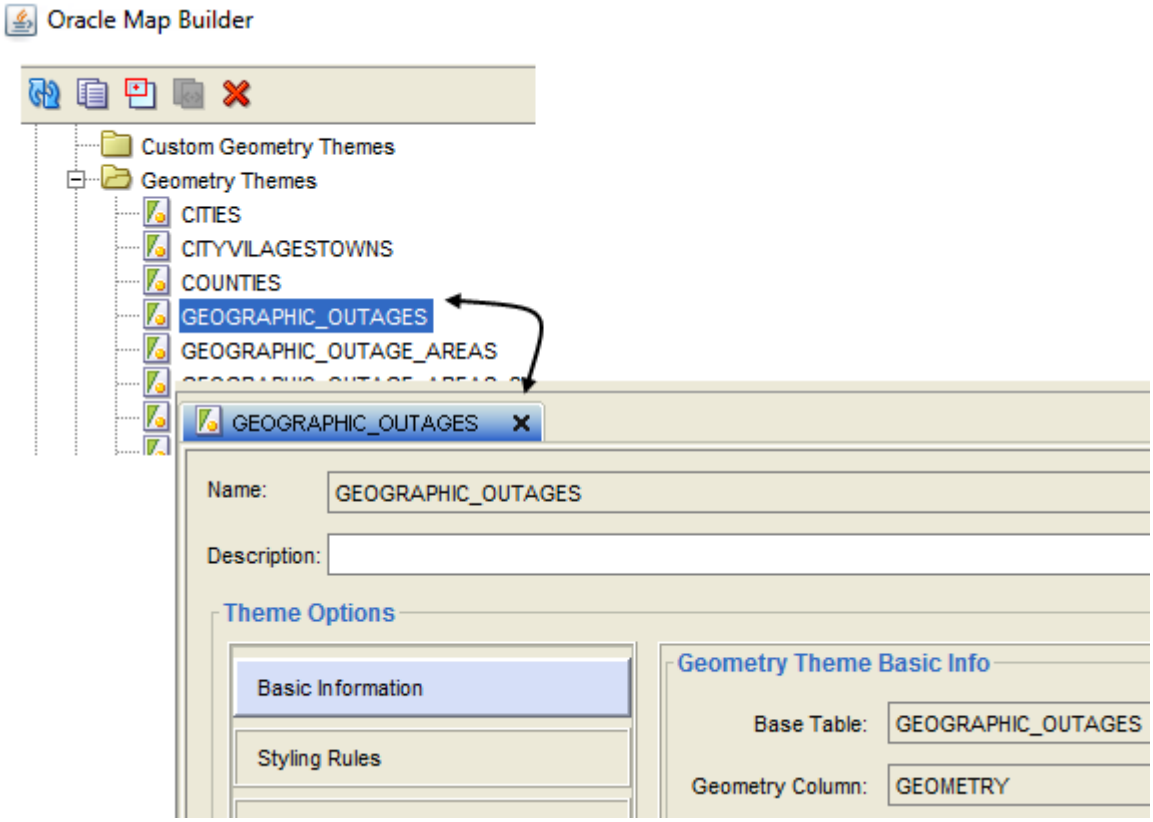


Prerequisites to supporting this functionality include:

1. Setting up the outage summary spatial tables in the NMS database. (See Preparing the NMS Model for Oracle Utilities Customer Self Service on page 8-148. for details) Oracle Utilities Customer Self (CSS) Service is not required to enable this outage summary in the NMS Viewer, however, the setup of the outage summary spatial table is required.
2. Create an Oracle Spatial MapBuilder Basemap containing and reference map themes (roads, water features, political boundaries) and include the spatial tables from the

previous set in themes. In the CSS setup process, you create a GEOGRAPHIC_OUTAGES table for outage location points and GEOGRAPHIC_OUTAGE_AREAS for outage summary by locations (*i.e.*, zip codes, cities, counties,…). You will want to make reference to one or both those tables in your map themes. In the image above, the viewer is showing outage locations as colored and sized dots and it is showing outage zip code areas as shaded polygons and annotating the number of customers out I the middle of the polygon.

In Map Builder, here is the definition of the GEOGRAPHIC_OUTAGES theme:



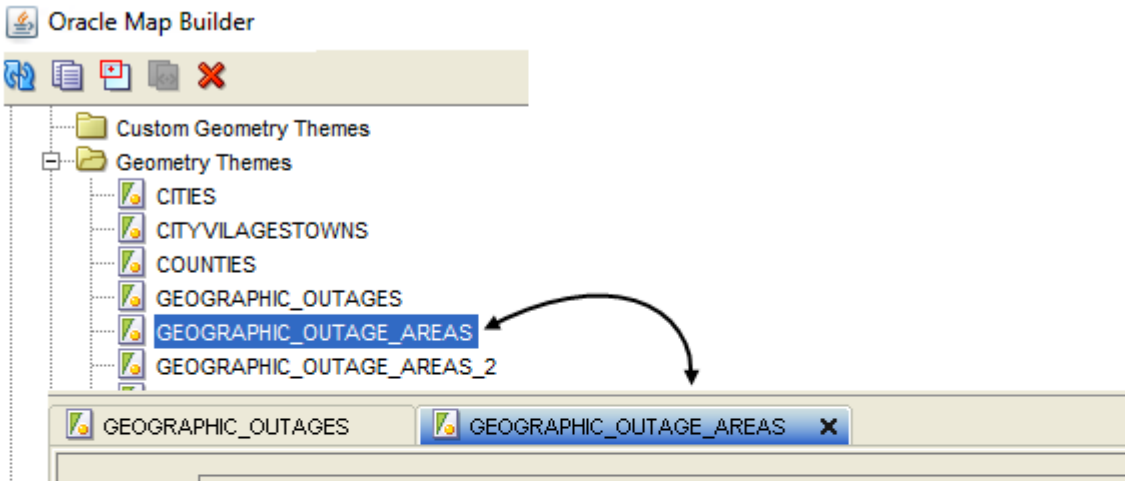
and the style rules to shade the polygons based on the cust_out and to label them with cust_out:

and here is the style rules for dots based on num_customers out:

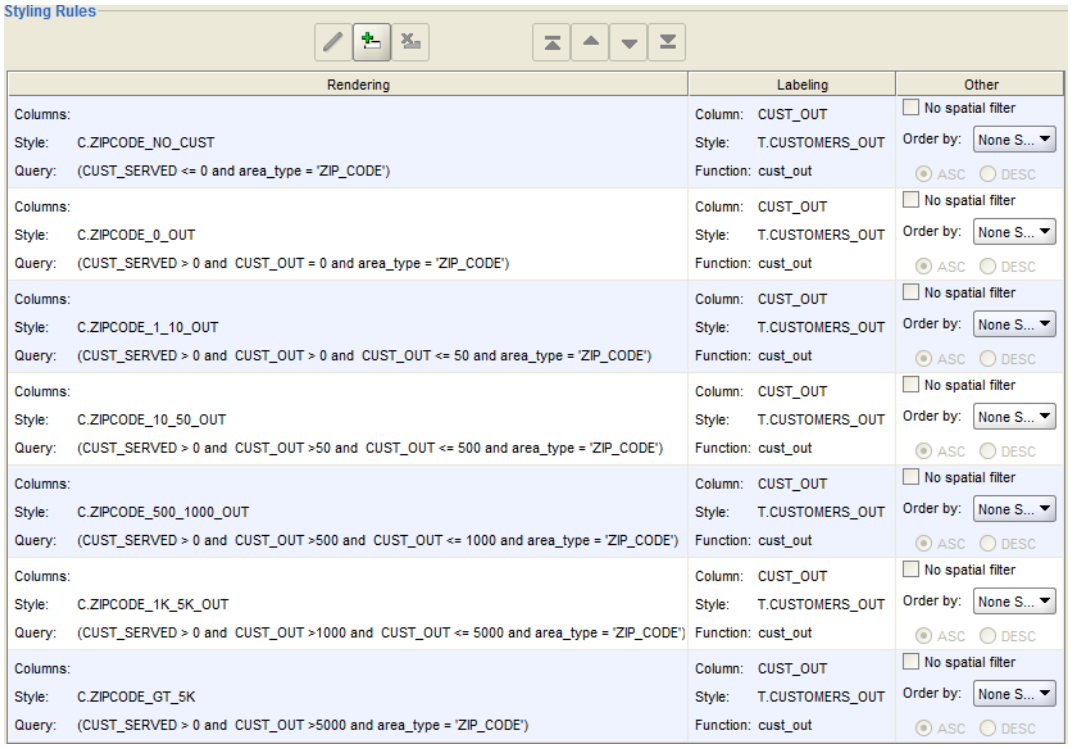
Styling Rules

Rendering	Labeling	Other
Columns: Style: M.OUTAGE_SM Query: (num_customers_out > 0 and num_customers_out <= 10)	Column: Style: Function: -1	No spatial filter Order by: None S... ASC DESC
Columns: Style: M.OUTAGE_MED Query: (NUM_CUSTOMERS_OUT > 10 and NUM_CUSTOMERS_OUT <=100)	Column: Style: Function: -1	No spatial filter Order by: None S... ASC DESC
Columns: Style: M.OUTAGE_LG Query: (NUM_CUSTOMERS_OUT >100 and NUM_CUSTOMERS_OUT <= 250)	Column: Style: Function: -1	No spatial filter Order by: None S... ASC DESC
Columns: Style: M.OUTAGE_XLG Query: (NUM_CUSTOMERS_OUT > 250 and NUM_CUSTOMERS_OUT <=1000)	Column: Style: Function: -1	No spatial filter Order by: None S... ASC DESC
Columns: Style: M.OUTAGE_HUGE Query: (NUM_CUSTOMERS_OUT > 1000)	Column: Style: Function: -1	No spatial filter Order by: None S... ASC DESC

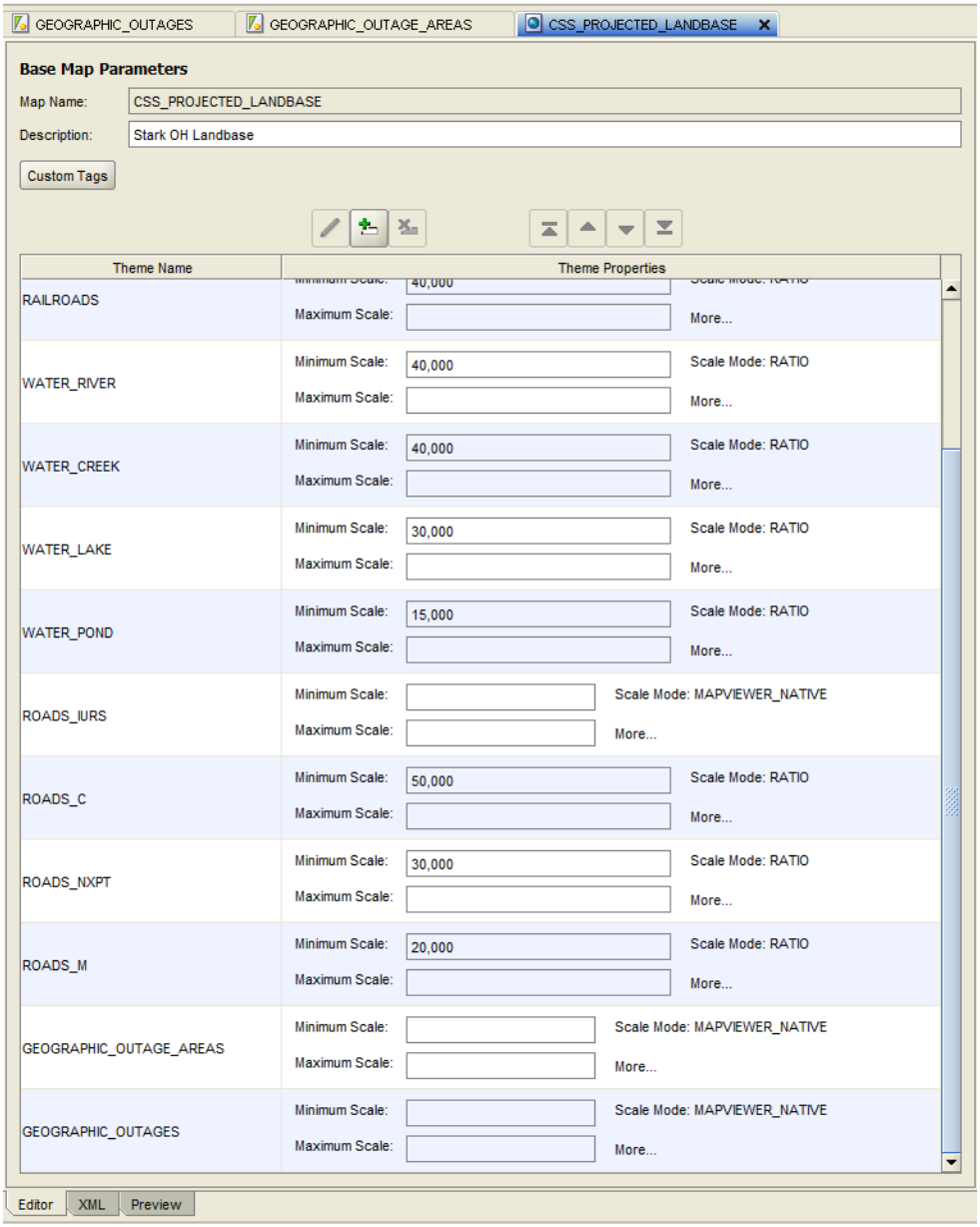
and here is the theme definition for the GEOGRAPHIC_OUTAGE_AREAS:



and the style rules to shade the polygons based on the cust_out and to label them with cust_out:



create a Basemap that incorporates one or both of the above themes:



3. Add your Oracle Spatial Basemap to the NMS Viewer Hide/Display configuration. This is described in the section "Oracle Spatial Server Connection - (Optional)" of the *Oracle Utilities Network Management System Installation Guide*.

OPAL has an example of this in its configuration. Follow the installation instructions in the *Oracle Utilities Network Management System Installation Guide* and include the following sections:

- Oracle Utilities Network Management System Validation Model Setup
- Oracle Spatial Server Connection - (Optional)
- Spatial Outage Summary Installation (Optional)

You will then have a running example of this functionality.

Model Build File Export to XML

NMS has two utilities for exporting model data files to XML. The resulting `.xml` file is defined by the `mb.xsd` XML schema file, which is found in `$NMS_BASE/product/sql/`.

MB Export to XML

To export an existing `.mb` file to XML, use the following command:

Usage:

```
mb2xml [-debug][-partitionClasses class_names ...]
    -classDirectory [class_directory_paths]
    -xml [xml-file] -mb [mb-file]
```

Options	Arguments	Description
`-debug`		enable debugging
`-partitionClasses`	*class_names*	additional partition class name(s)
-classDirectory	*class_directory_paths*	directories to find class file(s)
`-xml`	*xml-file*	output xml file name
`-mb`	*mb-file*	input mb file name

Note: Recall that the resulting `.xml` file is defined by the XML schema file: `$NMS_BASE/product/sql/mb.xsd`.

Schematic Data Export to XML

The schematic data file export takes a .mad file and exports it to XML.

Usage:

maa2xml [**-debug**] **-classDirectory** *class_directory_paths*
-xml *xml-file* **-maa** *maa-file*

Options	Arguments	Description
`-debug`		enable debugging
-classDirectory	*class_directory_paths*	directories to find class file(s)
`-xml`	*xml-file*	output xml file name
`-maa`	*maa-file*	input maa file name

Note: Recall that the resulting `.xml` file is defined by the XML schema file: `$NMS_BASE/product/sql/mb.xsd`.

Chapter 9

Database Maintenance

As general maintenance, you should establish a schedule to analyze tables, defragment your database, and purge historical/unnecessary data (then re-analyze the tables). You should also set up a schedule to backup your database and archive the backups.

This chapter describes all of these processes as well as the process of reconciling differences in database requirements when you upgrade your model to a new release of Oracle Utilities Network Management System.

It includes the following topics:

- Oracle Configuration
- Compatibility

Oracle Configuration

The following database settings are suggested for at least a minimum level of performance for an Oracle database. Any of these suggestions can be disregarded if an experienced Oracle DBA determines that other settings may offer better overall system performance. However, if any changes are made to any suggested parameters, performance of the system may be affected.

Indexes

Indexes should not be placed on the same physical disk as the data resides. If disk striping is being used then this requirement is not as critical, and may be ignored if enough disks are being employed.

Generating Statistics

As mentioned in a previous section of this chapter, tables should be analyzed periodically. The frequency can be determined by an experienced DBA, but it is suggested that this be done when the model or outage data changes substantially. This ensures that the Oracle statistics will be kept up to date for all of the database tables.

While most tables may be analyzed any time, some tables are cleared and repopulated while services are running. These tables should not be analyzed as part of a batch process; instead, they should be analyzed only while fully populated.

The following tables should only be analyzed **after** a full set of Feeder Load Management forecasts:

- flm_cap_bank_details
- flm_dev_violations_warnings
- flm_dist_gen_details
- flm_equiv_source_details
- flm_fdr_load
- flm_fdr_load_details
- flm_fdr_tie_points
- flm_feeders
- flm_islands
- flm_model_errors
- flm_reg_transfmr_details
- flm_solutions

Make Tablespaces Locally Managed

Dictionary managed tablespaces are more expensive on performance. It is suggested that the Oracle Utilities Network Management System tablespaces be setup as locally managed.

Block Size

The disk block size of the database should be the default of 8K, but could be set larger on recommendations from an experienced DBA.

Compatibility

An Oracle Utilities Network Management System schema is not backward compatible with Oracle Utilities Network Management System applications. Schema changes occur and are modified as the code and database move forward in time.

For example, it is unlikely that a database which has been migrated or built at version 1.7.10 code level will work with version 1.8.0 code level. However, data models are forward compatible, because Oracle Utilities Network Management System applications can migrate the database forward, making the necessary changes.

Thus, when backing up the database, you should note the Oracle Utilities Network Management System release level that was last operating against the database dump. That way, if there are other systems with older code, the data model is not imported into those systems and problems are not introduced.

Software

The Oracle Utilities Network Management System software is likely to be the most static data on the system. It should only be changing with upgrades. The need for software backup is generally low if the software is installed on several machines locally, but a weekly backup may be needed if there are maintenance scripts and SQL files being updated.

Map Files

Map files are replicated on a number of machines throughout the network, but they will change frequently. Data model files should be backed up once per week at minimum or nightly for frequently changing files.

Chapter 10

Site Guard

Oracle Site Guard can be used for switchover or failover from an NMS instance at one site (logical grouping of software components and associated hardware that run one or more user applications) to an NMS instance at another site (for instance, Disaster Recovery). It is also required for Failover Patching, which is a mechanism for installing patches with almost zero downtime described in chapter 3 of the *Oracle Utilities Network Management System Installation Guide*. Oracle Site Guard is a part of Oracle Enterprise Manager.

It includes the following topics:

- Prerequisites
- Site Installation and Configuration
- Custom Scripts
- Configuring Custom Scripts
- Operation Plans
- Additional NMS Configuration

Prerequisites

- Read *Site Guard Administrator's Guide* in the *Oracle Enterprise Manager Cloud Control Online Documentation Library* for your release of Oracle Enterprise Manager.
- Oracle Data Guard must be installed and configured on the Oracle RDBMS instances at each site, such that one site is Primary and the other site(s) are Standby. See the "Getting Started with Oracle Data Guard" chapter of *Oracle Data Guard Concepts and Administration*.

Optional, but recommended: Install the Application Management Pack for Oracle Utilities Network Management System in Oracle Enterprise Manager. Discover and configure the NMS targets at each site.

Site Installation and Configuration

Follow the instructions in the "Installing and Preparing Oracle Site Guard" chapter of *Site Guard Administrator's Guide* to prepare Oracle Site Guard, discover targets, and create a Generic System for each NMS site with the following targets:

- Database Instance
- Oracle WebLogic Domain
- Oracle Utilities NMS (if installed)
- Host targets that NMS services, WebLogic, and Database Instance run on.
- Any other targets for external adapters.

It is recommended that Oracle Utilities NMS targets are made the key members of the system and that Availability Criteria is set to "Any Of The Key Members". With this configuration, the system will be shown as available if either of the NMS instances in Dual-Environment Configuration is available.

Continue following the instructions in the "Installing and Preparing Oracle Site Guard" chapter of *Site Guard Administrator's Guide* to create credentials.

Follow the instructions in the "Configuring Oracle Site Guard" chapter of *Site Guard Administrator's Guide* to create configuration for Primary and Standby sites and to associate credentials with each generic system.

Custom Scripts

Oracle Site Guard allows for custom scripts to be used during operations to perform tasks not natively supported by Site Guard.

NMS provides templates for Site Guard custom scripts in `$NMS_BASE/templates`. They must be installed in a common location on each NMS server or each database server, depending on the script. Scripts should be installed with `.template` removed from their names and made executable unless otherwise noted. Some of these scripts must be edited and others can be edited.

Custom Scripts for Database Server

This must be configured to run as the user that is running the database.

sg-createrestorepoint.template

Shell script that creates a guaranteed restore point in the database named "BEFORE_PATCH_NMS". If `$HOME/.profile` does not set `ORACLE_HOME`, `ORACLE_SID`, and `PATH` such that "sqlplus / as sysdba" connects to the database, then this script must be edited to set those environment variables.

sg-createrestorepoint.sql.template

SQL script used by `sg-createrestorepoint.template`. Does not need to be executable.

Custom Scripts for NMS server

These must be configured to run as the NMS administrative user. When Dual-Environment Configuration is used, these scripts must be configured twice – once for each environment.

sg-nmsreadonly.template

Shell script that calls `nms-read-only` to turn on Read Only mode.

sg-nmsstart.template

Shell script that calls `nms-all-start` to start NMS services and the WebLogic managed server.

sg-nmsstop.template

Shell script that calls "`nms-all-stop -fast`" to stop NMS services and the WebLogic managed server.

sg-nmsswitchover.template

Shell script which handles a switchover. Run from the Standby site that is becoming Primary. This:

- Syncs the operations model from the old primary site to the standby site that is becoming primary
- Updates the NMS schema at the standby site to indicate that it is the current primary site
- Starts NMS Services and the WebLogic managed server
- Creates a system alarm indicating that this is the new primary site.

This script must be customized to match the sites and hosts at which NMS is installed. See the comments in the template.

Other Custom Scripts

Other custom scripts can be added for any other business needs.

Configuring Custom Scripts

Follow the instructions in the "Configuring Oracle Site Guard" chapter of *Site Guard Administrator's Guide* to configure the Pre Scripts and Post Scripts shown below, using the shown parameters for Script Type, Operation, and Role and running on the given target host. Scripts must be configured to run as the user specified in the Custom Scripts section.

Script	Target Host	Script Type	Operation	Role	Notes
sg-nmsstart	NMS Server	Post-Script	Start	N/A	
sg-nmsstop	NMS Server	Post-Script	Stop	N/A	
sg-nmsstop	NMS Server	Post-Script	Switchover	Primary	
sg-nmsswitchover	NMS Server	Post-Script	Switchover	Standby	
sg-nmsswitchover	NMS Server	Post-Script	Failover	Standby	Only used by Failover
sg-nmsreadonly	NMS Server	Pre-Script	Failover	Primary	Only used by Failover Patching
sg-createrestorepoint	Database Server	Pre-Script	Failover	Primary	Only used by Failover Patching

Operation Plans

These are the Site Guard operation plans for NMS. Follow the instructions in the Creating Operation Plans section of the "Performing Oracle Site Guard Operations" chapter of *Site Guard Administrator's Guide* to create the operation plans you will be using. Choose the Operation Type specified under the Operation Type heading for each operation. For operations only involving one site (Start and Stop), the operation must be configured for each site. For operations involving two sites (Switchover, Failover, and Failover Patching), an operation plan must be created for each combination of Primary and Standby site. For example, with three sites (A, B, and C), these combinations must be configured: A → B, A → C, B → A, B → C, C → A, and C → B. After an operation plan is created, edit it according to the instructions in the Operation Configuration section. See the Editing and Updating Operation Plans section of the "Performing Oracle Site Guard Operations" chapter of *Site Guard Administrator's Guide*.

Start Operation

Operation Type

Start

Description

This operation starts NMS services and the WebLogic managed server at a site.

Operation Configuration

Disable the Start Managed Server steps in the operation plan.

Stop Operation

Operation Type

Stop

Description

This operation stops NMS services and the WebLogic managed server at a site. The WebLogic Node Manager and AdminServer are left running.

Operation Configuration

Disable the Stop Domain, Stop Managed Server, Stop Admin Server, and Stop Node Manager steps in the operation plan.

Switchover Operation

Additional Prerequisites

- SSH password-less login must be enabled between the NMS servers on each site.

Operation Type

Switchover

Description

This operation shuts down NMS at the primary site, performs a switchover operation in the Oracle RDBMS to make the chosen standby site primary, and starts NMS there. The original primary site becomes a standby site.

Operation Configuration

Disable the Stop Domain, Stop Managed Server, Stop Admin Server, Stop Node Manager, and Start Managed Server steps in the operation plan.

Failover Operation

Additional Prerequisites

- Oracle Flashback Database must be enabled in Oracle RDBMS at all sites.
- SSH password-less login must be enabled between the NMS servers on each site.

Operation Type

Failover

Description

This operation performs a failover operation in the Oracle RDBMS to make the standby database primary and starts NMS at the standby site. This operation can be executed even if the primary site is not reachable. After successful completion, both the primary and standby site will be primary. NMS must be stopped at the original primary site and the Oracle RDBMS instance reinstated as standby

Operation Configuration

Disable the Start Managed Server step in the operation plan.

Delete the Run Script steps for the Pre-Scripts `sg-nmsreadonly` and `sg-createrestorepoint` in the operation plan.

Failover Patching Operation

Additional Prerequisites

- Oracle Flashback Database must be enabled in Oracle RDBMS at all sites.
- SSH password-less login must be enabled between the NMS servers on each site.
- Dual-Environment Configuration must be used on all sites. See Chapter 6 - Environment Configuration for information.

Operation Type

Failover

Description

This operation

- Creates a guaranteed restore point in the primary Oracle RDBMS.
- Turns on Read Only mode for NMS users at the primary site.
- Performs a failover operation in the Oracle RDBMS to make the standby database primary.

This operation is used as part of the **Steps to Deploy a Patch Bundle using Failover Patching** section in Chapter 3 of the *Oracle Utilities Network Management System Installation Guide* and is not meant to be used outside of this context.

Operation Configuration

Enable the Run Script steps for the Pre-Scripts `sg-nmsreadonly` and `sg-createrestorepoint` and set the Error Mode to Stop on Error in the operation plan.

Delete the Run Script step for the Post-Script sg-nmsswitchover in the operation plan.

Additional NMS Configuration

Read Only Mode

Read Only mode disables buttons and menu items in the Java Application which save data. This is used as part of Failover Patching to prevent users from saving data that will be lost. This is controlled by the FLG_READ_ONLY status flag in JBot configuration. Custom buttons and menu items added to JBot configuration should use the FLG_READ_ONLY status flag to control whether they are enabled.

Example

```
<Enabled when= "!FLG_READ_ONLY "/>
```

Applications which have a status bar (Web Workspace, Model Management, Web Callbacks, and Storm Management) display a read only mode icon in the status bar. This is configured in the LBL_READ_ONLY label in each application's status bar.

In addition, the title bar of each java application changes while in Read Only mode. The title is configured with the [Application].readOnlyTitle property in each application. The default configuration adds "(Read Only mode)" to the original application's title.

Read Only mode can be turned on by running the following command:

```
nms-read-only 1
```

And turned off with the following command:

```
nms-read-only 0
```

When Read Only mode is turned on or off, a system alarm is generated using the text configured for the READ_ONLY_ON and READ_ONLY_OFF message codes in the MESSAGE_CODE_LOOKUP table for MESSAGE_TOPIC 'DDS'.

Example

```
INSERT INTO message_code_lookup (message_topic, message_code,
                                 message_string)
VALUES ('DDS', 'READ_ONLY_ON', 'Read Only mode turned on');
INSERT INTO message_code_lookup (message_topic, message_code,
                                 message_string)
VALUES ('DDS', 'READ_ONLY_OFF', 'Read Only mode turned off');
```

Syncing Operations Model

In order for the Operations Model to be up to date when a switchover or failover is performed, the `$OPERATIONS_MODELS` directory is synced to the standby site. However, this can involve a lot of data and in the case of a failover, the original primary site might not be reachable. The `nms-sync-site` script should be used to keep standby sites in sync with the primary site. Create the file `$NMS_HOME/etc/standby_sites.txt` on each NMS environment with a list of the hostnames where NMS services run at each of the other sites (one per line).

In the `[project]-postbuild` script, add the following line after all updates to `$OPERATIONS_MODELS` are finished:

```
nms-sync-site -f model-build @$NMS_HOME/etc/standby_sites.txt
```

In addition, it is recommended to create a cron table entry on each environment to run the following script:

```
#!/bin/ksh
cnt=$(ISQL -silent << sqlend
set pagesize 0
set feedback off
select count(*) from nms_current_site
where site='$NMS_SITE_NAME';
exit;
sqlend
)
if [ $cnt -gt 0 ] && nms-version --is-active
then
   nms-sync-site -f model-build @$NMS_HOME/etc/standby_sites.txt
fi
```

nms-wls-config

Since `nms-all-start` and `nms-all-stop` are used in various custom scripts, `nms-wls-config` must be run in each environment to configure starting/stopping the WebLogic managed server. See the Starting and Stopping Services on page 7-20 for more information.

Chapter 11

High Availability Configuration

Overview

The High Availability module is an optional module consisting of a group of components that together combine to facilitate the continual monitoring and automatic disaster recovery of NMS sites. The high availability module provides:

- Continual monitoring of the state of each critical component within each NMS site
- Automatic disaster recovery from one NMS site to another.
- A comprehensive set of web pages for reporting the state of each site and its components and configuration of the sites being monitored.

The main components involved in the automatic failover and recovery process are defined below.

NMS Agent

The NMS Agent runs on each NMS Services server and is responsible for monitoring NMS back-end services and reporting their state to the NMS Monitor module. In a dual-environment configuration, both administrative users will have an NMS Agent instance.

NMS Monitor

The NMS Monitor runs on a WebLogic managed server at each site and is responsible for periodically requesting the current status of the site from the NMS Agents, WebLogic managed servers and databases, storing the status of the sites in a ZooKeeper cluster, coordinating with other NMS Monitors in the system, determining the state of each site and triggering automatic failover and recovery of the system through Site Guard.

CESEJB, NMS-WS

The CESEJB and NMS-WS deployments provide REST APIs to allow the NMS Monitor to request the current status of the deployment.

WebLogic Server

The WebLogic servers are used to deploy the NMS Monitor. An NMS Monitor managed server is installed on each WebLogic instance. The WebLogic Admin Server reports the state of all managed servers and their applications to the NMS Monitor.

Database Server

The NMS Monitor interrogates the databases to determine the active and staging environment for each site.

EMCLI

The Enterprise Manager Command Line Interface (emcli) is installed on each WebLogic server and is used by the NMS Monitor to read site and operation plans from the Site Guard and request the automatic execution of a failover operation plan.

Site Guard

Oracle Site Guard is used for disaster recovery to initiate failover from an NMS instance at one site to an NMS Instance at another site. For full details of Site Guard and its configuration, see Site Guard.

ZooKeeper

An Apache ZooKeeper cluster is used to store the site status information collated by each of the NMS Monitors allowing each monitor to see the complete state of all NMS sites in the system in order to make an intelligent decision on the best disaster recovery plan to execute.

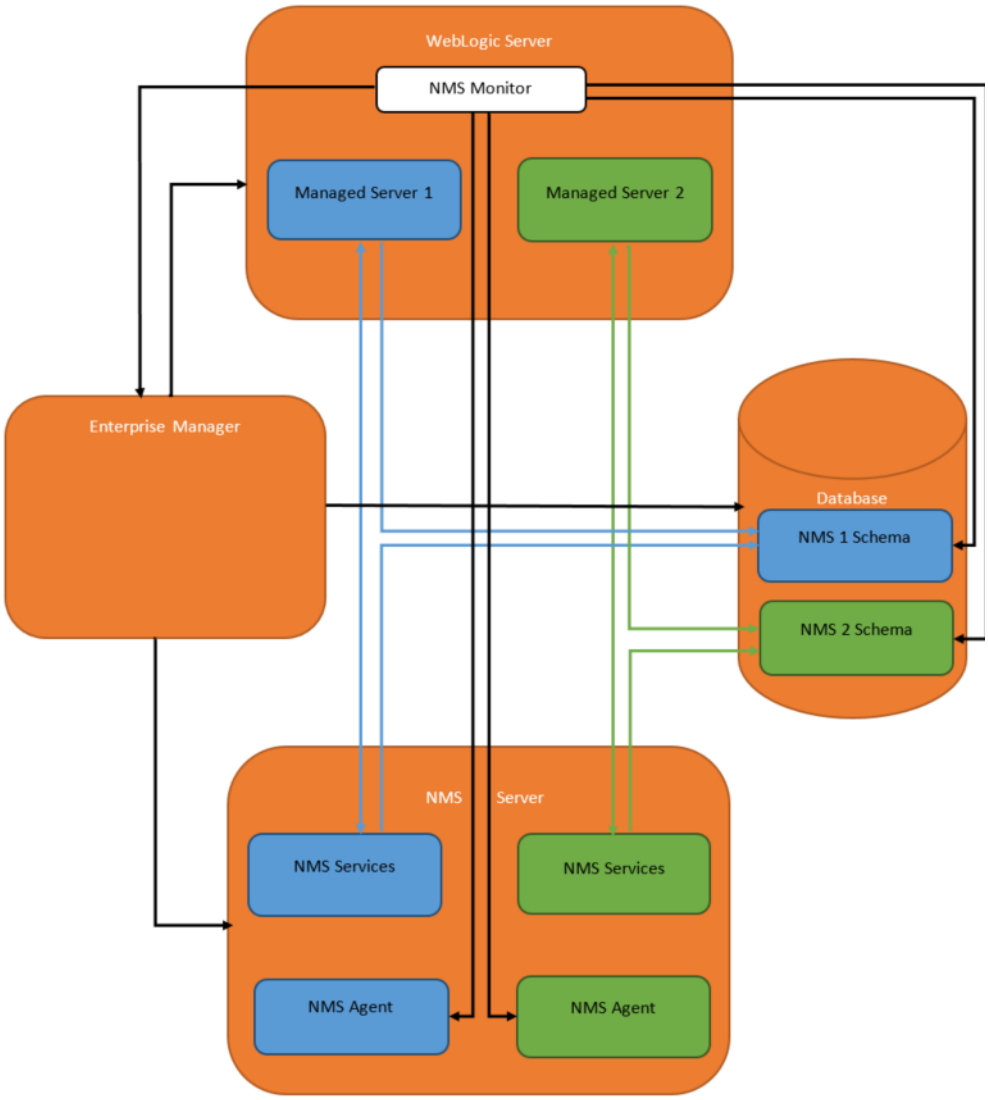
Architectural Overview

Components at One NMS Site

The following diagram outlines the components required at one NMS Site in order to support the High Availability module. The site employs a dual “blue green” environment where blue represents one environment and green the other.

The NMS Monitor is deployed as a managed server on the WebLogic server and communicates with the NMS Agent and Database of each environment.

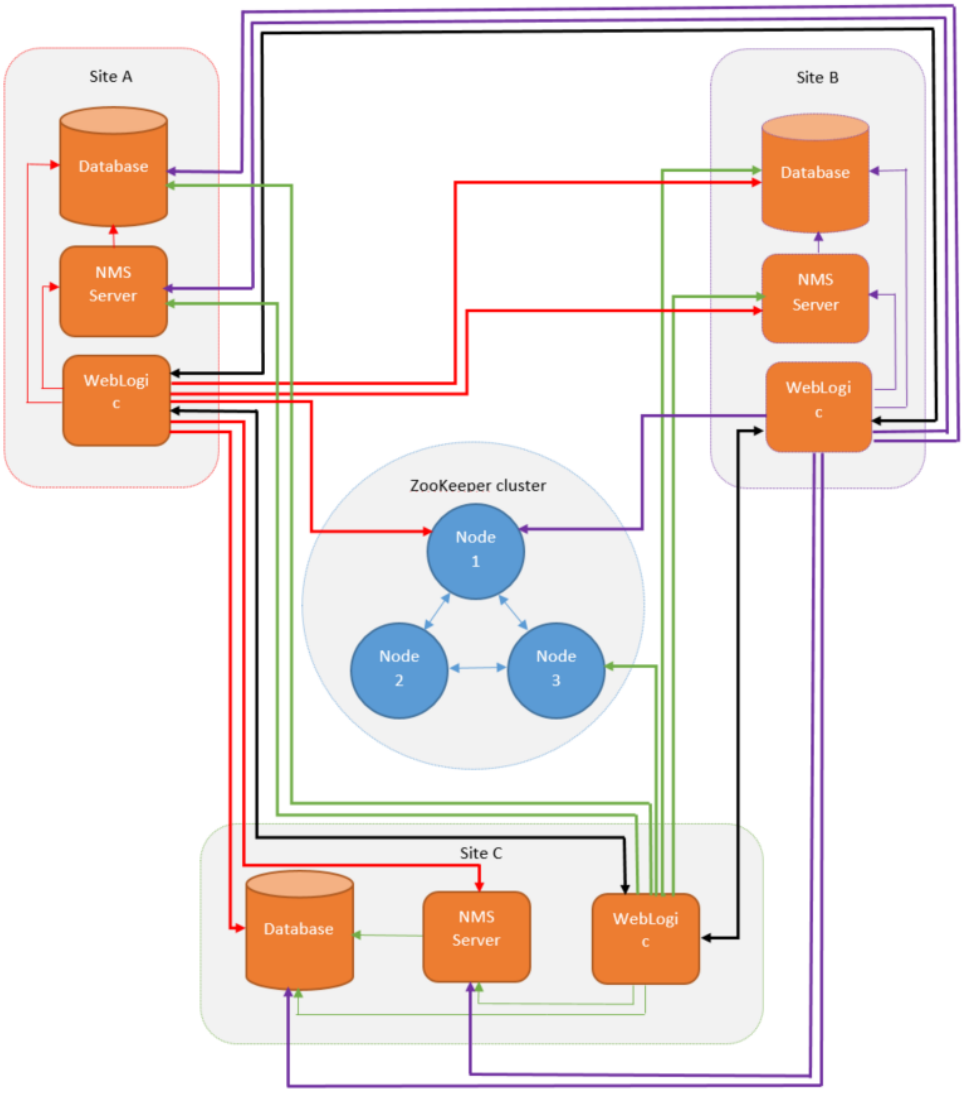
Each environment has its own NMS Agent who monitors the NMS Services in that environment.



Components at Three Sites

The following diagram outlines the components at three NMS Sites required in order to support the High Availability module.

The NMS Monitor deployed in each sites WebLogic server has a connection to a node in the ZooKeeper cluster.



Recommended Configuration Order

The following sections outline the steps necessary to configure the NMS Monitors, NMS Agents and ZooKeeper. However there are some configuration dependencies between each of these components. Specifically, the identity stores and truststores of the ZooKeeper, NMS Monitor, and NMS Agent nodes need to exist before each other's certificates can be imported. Below is a recommended order when configuring these node:

- ZooKeeper
  - Install ZooKeeper on each node
  - Create the identity store on each node, see ZooKeeper -> Keystore creation
  - Export the certificate on each node, see ZooKeeper -> Export the certificate
  - Create the truststore store on each node and import the certificates from each of the other nodes, see ZooKeeper -> Creating the truststores
  - Configure the environment variables for each node, see
  - ZooKeeper -> Environment
  - Generate the ZooKeeper configuration file, see ZooKeeper -> ZooKeeper configuration file
  - Create the ZooKeeper data directory, see ZooKeeper -> ZooKeeper data directory
- NMS Monitor
  - Download and install the Enterprise Manager Command Line Interface (EMCLI), see EM CLI Standard.
  - Set the WebLogic domain environment in a terminal, see NMS Monitor -> Set the domain environment
  - Configure the connection to the ZooKeeper cluster, see NMS Monitor -> Configure ZooKeeper
  - Configure the NMS datasources, see NMS Monitor -> Configure Datasource
  - Configure the location of the EMCLI, see NMS Monitor -> Configure emcli, Set home.
  - Set the keystore location for each NMS Monitor, see NMS Monitor -> Keystore actions -> Set keystore location
  - Create the keystore for each NMS Monitor, see NMS Monitor -> keystore actions -> Create keystore
  - Export the certificate for the NMS Monitor, see NMS Monitor -> keystore actions -> Export certificate
  - Set the truststore location for each NMS Monitor, see NMS Monitor -> Truststore actions -> Set truststore location
  - Create the truststore for each NMS Monitor, see NMS Monitor -> Truststore actions -> Create truststore
  - Create the credentials for the emcli, see NMS Monitor -> Credentials -> Create credentials.
  - Create the credentials for the WebLogic admin (#1), see NMS Monitor -> Credentials -> Create credentials
  - Create the credentials for the NMS cesejb managed server 1 (#2), see NMS Monitor -> Credentials -> Create credentials

    - Create the credentials for the NMS cesejb managed server 2 (#3), see Credentials -> Create credentials
    - Create the credentials for the NMS OMA managed server 1 (if different from the CESEJB Managed server) (#4), see NMS Monitor -> Credentials -> Create credentials
    - Create the credentials for the NMS OMA managed server 2 (if different from the CESEJB Managed server) (#5), see NMS Monitor -> Credentials -> Create credentials
    - Create the NMS Monitor server in WebLogic, see NMS Monitor -> Creating the NMS Monitor server
- NMS Agent
    - Configure the server properties on each NMS Agent node, see NMS Agent -> Configure Server
    - Set the keystore location for each NMS Agent, see Keystore actions, NMS Agent -> Set keystore location
    - Create the keystore for each NMS Agent, see NMS Agent -> Keystore actions -> Create keystore
    - Export the certificate for the NMS Agent, see NMS Agent -> Keystore actions -> Export certificate
    - Set the truststore location for each NMS Agent, see NMS Agent -> Truststore actions, -> Set truststore location
    - Create the truststore for each NMS Agent, see NMS Agent -> Truststore actions -> Create truststore
    - Run each NMS Agent as a service, see NMS Agent -> Generate service script
    - Create the nmsagent directory on each NMS Agent, see NMS Agent -> The nmsagent directory
- ZooKeeper
    - Import each of the NMS Monitor certificates into each ZooKeeper nodes truststore, See ZooKeeper -> Creating the truststores
- NMS Monitor
    - Import each of the ZooKeeper node certificates into each of the NMS Monitors truststores, see NMS Monitors -> Truststore actions -> Import certificate
    - Import each of the NMS Agent node certificates into each of its associated NMS Monitor truststores, see NMS Monitors -> Truststore actions -> Import certificate
    - Import the NMS self-signed certificate into the java truststore (optional), see NMS Monitors -> NMS CESEJB Self-Signed Certificates
- NMS Agent
    - Import each of the NMS Monitor certificates into its associated NMS Agent truststore, see NMS Agent -> Truststore actions -> Import certificate
- NMS Monitor Web Application
    - Configure the Site Configuration within the NMS Monitor web application, see NMS Monitor Web Application -> Configuration Screens -> Sites
        - Add each of the sites to be monitored
        - For each site
            - Add a URL for the NMS Agent in environment 1 into Agent1.
            - Add a URL for the NMS Agent in environment 2 into Agent2.

    - Add a URL for the WebLogic administration server. This must match the credentials entered in NMS Monitor properties (See #1)
    - Add a URL for the first NMS CESEJB managed server into NMS Env 1. This must match the credentials entered in NMS Monitor properties (See #2)
    - Add a URL for the second NMS CESEJB managed server into NMS Env 2. This must match the credentials entered in NMS Monitor properties (See #3)
    - Add a URL for the first NMS OMA managed server into OMA Env 1. This must match the credentials entered in NMS Monitor properties (See #4)
    - Add a URL for the second NMS OMA managed server into OMA Env 2. This must match the credentials entered in NMS Monitor properties (See #5)
- Configure the automatic failover properties, see NMS Monitor Web Application -> Configuration Screens -> Automatic Failover
- Configure the failover priority, see NMS Monitor Web Application -> Configuration Screens -> Failover Priority
- Configure the failover operation plans, see NMS Monitor Web Application -> Configuration Screens Failover operation plans

EM CLI Standard

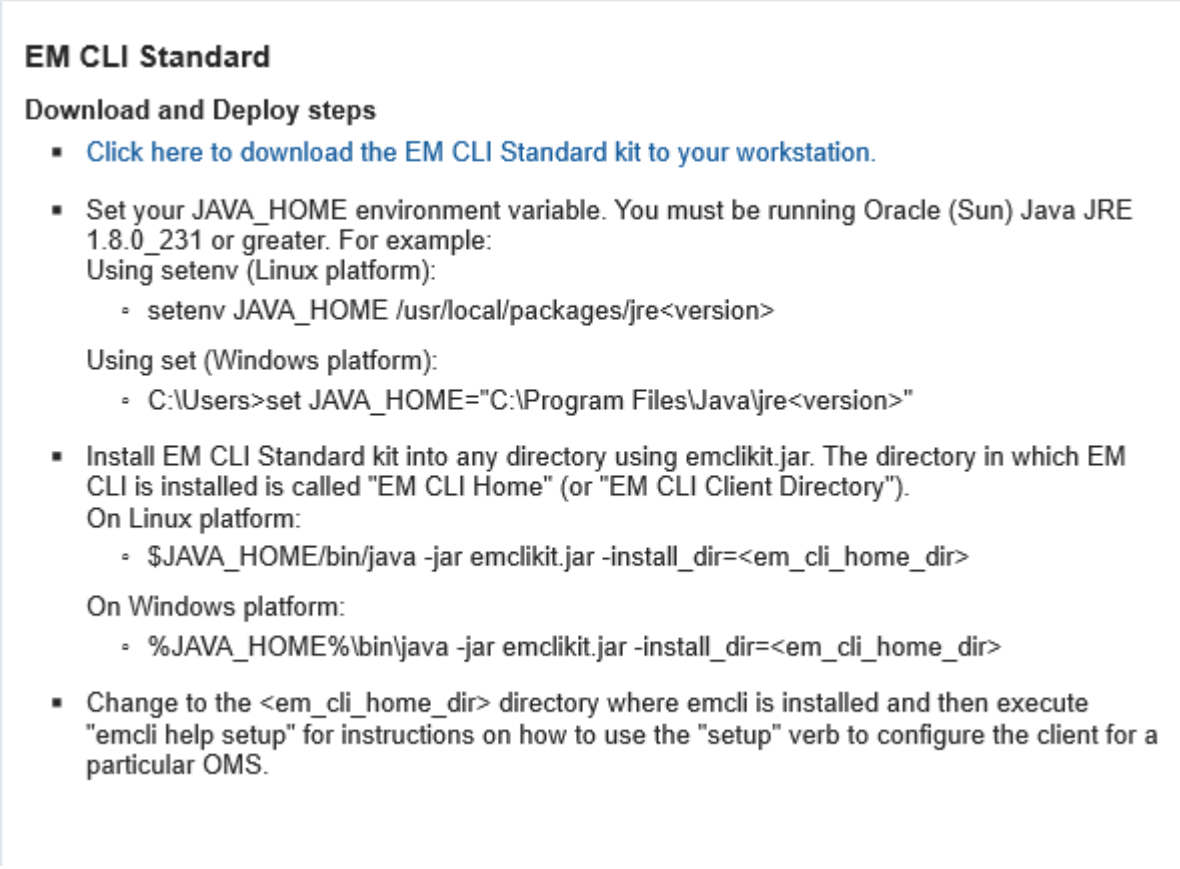
The Enterprise Manager Command Line Interface (EM CLI) enables users to access Enterprise Manager functionality through a command-line interface or scripts. This is used by the NMS Monitor to communicate with the Enterprise Manager in order to read site operation plans and to request failover between sites.

Download

The CLI can be downloaded from an installed enterprise manager from the main page by navigating to Setup -> Command Line Interface or by using the following URL:

```
https://<hostname>:7803/em/faces/core-emcli-emcliDownload
```

Where <hostname> is the host that the Enterprise Manager is installed on.



The web page carries a link to the emclikit.jar which should be downloaded to each server running NMS Monitor.

Installation

In order to install the CLI, you must first create its home directory. In this example we will may the CLI home directory $ HOME/emcli. This assumes that the emclikit.jar file has previously been downloaded to the $NMS_HOME directory. Issue the following commands:

```
cd $ HOME
mkdir emcli
java –jar emclikit.jar –install_dir=emcli
```

Setup

Once the CLI has been installed it must be setup. Issue the following commands to setup the CLI:

```
cd emcli
emcli setup -url=https://<hostname>:7803/em -username=<username>
```

where <hostname> is the name of the server on which the Enterprise Manager is deployed and <username> is the administrator user name for the Enterprise Manager.

Configuring NMS Monitor

Each NMS Monitor instance must be configured to use the CLI, specifically the CLI home and the credentials used to login to the Enterprise Manager must be configured. More details on this can be found in the NMS Monitor chapter.

ZooKeeper

NMS Monitor uses an Apache ZooKeeper cluster as its database. It is recommended the ZooKeeper cluster consist of at least 3 nodes to allow 1 node to be brought down for backup while still maintaining redundancy.

In this chapter it is assumed that the cluster is configured on 3 servers Server1, Server2, Server3 and that ZooKeeper is pre-installed in the home directory of the user.

ZooKeeper can be downloaded from the following location: https://zookeeper.apache.org/releases.html

The following instructions are based on ZooKeeper 3.6.2.

Keystore Creation

In order to support SSL between nodes in the cluster and between ZooKeeper and clients each node must have its own keystore and trust store configured. These should be put into a new directory called ssl:

```
cd $HOME/apache-zookeeper-3.6.2-bin
mkdir ssl
cd ssl
```

Use the following command to create the identity keystore and create the server certificate:

```
keytool -genkey -keyalg RSA -keystore ./keystore.p12 -alias
<hostname>  -ext "SAN:c=DNS:<hostname>,IP:<ipaddress>" -storetype
pkcs12
```

where <hostname> is the name of the server and <ipaddress> is the ip address of the server.

This action should be performed on each node in the cluster.

The keystore can be either in jks or pkcs12 format. All examples here are in pkcs12 format. Depending on the format the keystore or trustores created must end with the .jks or .p12 suffix.

Export the Certificate

Use the following command to export the node's certificate. This will be imported into the truststore of the clients and other cluster nodes.

```
keytool -export -alias <hostname> -rfc -keystore ./keystore.p12 >
./<hostname>.cert
```

where <hostname> is the name of the server.

This action should be performed on each node in the cluster.

Creating the Truststores

On each server a truststore must be created in the $HOME/ apache-zookeeper-3.6.2-bin/ssl directory. It should be created and all client certificates imported into the truststore in addition to the certificate from each node in the cluster.

When the first certificate is imported, the truststore will automatically be generated if it does not exist.

Use the following command to import a certificate and create the truststore.

```
cd $HOME/apache-zookeeper-3.6.2-bin/ssl keytool -import -alias
<nodename> -file <nodename.cert> -keystore ./truststore.p12 -storetype
pkcs12s
```

Where <nodename> is the name of the node in the cluster, or the client name and <nodename.cert> is the location of the certificate to be imported.

Environment

In order to support SSL between nodes in the cluster and between ZooKeeper and clients the following environment variables must be set in the user's profile on each ZooKeeper node.

```
SERVER_JVMFLAGS="
-
Dzookeeper.serverCnxnFactory=org.apache.zookeeper.server.NettyServerCn
xnFactory
-Dzookeeper.ssl.keyStore.location=/scratch/gbuora/rdj/apache-
zookeeper-3.6.2-bin/ssl/keystore.p12
-Dzookeeper.ssl.keyStore.password=password
-Dzookeeper.ssl.trustStore.location=/scratch/gbuora/rdj/apache-
zookeeper-3.6.2-bin/ssl/truststore.p12
-Dzookeeper.ssl.trustStore.password=password
-Dzookeeper.ssl.quorum.keyStore.location=/scratch/gbuora/rdj/apache-
zookeeper-3.6.2-bin/ssl/keystore.p12
-Dzookeeper.ssl.quorum.keyStore.password=password
-Dzookeeper.ssl.quorum.trustStore.location/scratch/gbuora/rdj/apache-
zookeeper-3.6.2-bin/ssl/truststore.p12
-Dzookeeper.ssl.quorum.trustStore.password=password
-Dzookeeper.ssl.quorum.hostnameVerification=false"
```

The password should be set to the password used when generating the identity keystore or truststore. Likewise the names of the identity store and truststore should refer to the ones created earlier in this chapter.

ZooKeeper Configuration File

The directory `$HOME/apache-zookeeper-3.6.2-bin/conf` contains the configuration file for the ZooKeeper node. The file `zoo.cfg` should be created in this directory on each node.

Below is an example of the configuration file:

```
# The number of milliseconds of each tick
tickTime=2000
# The number of ticks that the initial
# synchronization phase can take
initLimit=10
# The number of ticks that can pass between
# sending a request and getting an acknowledgement
syncLimit=5
# the directory where the snapshot is stored.
dataDir=/scratch/zookeeper
# secure client port
secureClientPort=2281
# secure quorum
sslQuorum=true
server.1=<server1_hostname>:2888:3888
server.2=<server2_hostname>:2888:3888
server.3=<server3_hostname>:2888:3888
```

The `tickTime`, `initLimit` and `syncLimit` are set to default values.

The `datadir` must be set to the location the ZooKeeper data will be stored in (see below).

The `secureClientPort` defines the SSL port number that will be used by clients to connect to the ZooKeeper instance (See the chapter on NMS Monitor for details of how the client is configured to connect to the ZooKeeper cluster).

The sslQuorum property enables SSL communication between nodes in the ZooKeeper cluster.

The following lines:

```
server.1=<server1_hostname>:2888:3888
server.2=<server2_hostname>:2888:3888
server.3=<server3_hostname>:2888:3888
```

Define each of the nodes that make up the ZooKeeper cluster, where <server1_hostname>, <server2_hostname>, <server3_hostname> are the host names of each node in the cluster. The nodes will use ports 2888 and 3888 to communicate with each other.

Node Id

In this example, each node is assigned a unique id of `1`, `2`, or `3` depending on the server property

```
server.2=<server2_hostname>:2888:3888
```

Assigns the id 2 to host <server2_hostname>

ZooKeeper Data Directory

The data directory needs to be created on each node of the server. The location is configured in the dataDir property in the ZooKeeper configuration file (see above).

The data directory must be created prior to starting the ZooKeeper instance and the file 'myid' must exist in the directory. The contents of the myid file must be the id of the node. For example, to configure the data directory for a server whose node is 2 and whose data location is `$HOME/zookeeper`:

```
cd $HOME
mkdir zookeeper
cd zookeeper
echo 2 > myid
```

Starting and Stopping the Node

In order to start the ZooKeeper node use the following command:

```
apache-zookeeper-3.6.2-bin/bin/zkServer.sh start
```

To stop the node use the following command:

```
apache-zookeeper-3.6.2-bin/bin/zkServer.sh stop
```

NMS Monitor

Overview

The NMS Monitor runs on a WebLogic managed server at each site and periodically request the current status of the NMS Agents, WebLogic managed servers and databases. The monitor is configured to communicate with an Oracle Enterprise Manager instance to trigger a failover request via Site Guard in the event of loss to a critical component. Requests for failover are sent to the Site Guard instance via the Enterprise Manager Command Line Interface (emcli).

Each NMS Monitor in the system communicates with an Apache ZooKeeper cluster in order to:

- manage and read the site configuration
- store the current site status
- coordinate with other NMS Monitor instances

In this way the NMS Monitor is able to determine the state of all sites in the system.

The NMS Monitor hosts a web page showing the current status at each site and provides configuration to allow the addition and maintenance of sites to be monitored.

Monitoring

The NMS Monitor monitors the following components using REST API calls and stores the status of each in the ZooKeeper cluster:

- From each NMS Agent the monitor retrieves the current and previous state of the services, timestamps and the NMS version.
- From the CESEJB service the monitor retrieves the status of the services and publisher.
- From the Mobile NMS service the monitor retrieves the status of the services.
- From the databases the monitor retrieves the active and staged environments, active and staged versions and the site name.
- From the WebLogic instance the monitor retrieves the status of each managed server and the applications that are deployed under it.

The NMS Monitor uses this information when determining:

- Whether the active NMS instance is fully running
- Which standby instances are viable candidates for failover.

Configuration

The NMS Monitor is configured via the nmmonitor.properties configuration file located in the `$DOMAIN_HOME` directory of the WebLogic instance. A configuration tool, nms-monitor-config is provided to help in the configuration of the monitor. See NMS Monitor for full details of the configuration of the monitor and the use of the nms-monitor-config tool.

Setting the Domain Environment

The WebLogic domain environment must be sourced prior to running the nms-monitor-config utility, for example:

```
cd $DOMAIN_HOME/bin
. ./setDomainEnv.sh
```

Configuration Tool Main Page

The main page displays the following options:

1 : Display properties

2 : Configure zookeeper

3 : Configure datasource

4 : Configure emcli

5 : Keystore actions

6 : Truststore actions

7 : Credentials

0 : Exit

Display properties: Selecting this option displays the current configuration properties for the NMS Monitor.

Configure ZooKeeper: This option configures the properties necessary to connect to the ZooKeeper cluster.

Configure datasource: This option is used to identify the database datasources configured in the WebLogic server that are used to connect to the NMS database.

Configure emcli: This option is used to configure the properties for the Enterprise Manager command line interface.

Keystore Actions: This options allows the operator to create an identity keystore for the monitor and export the monitor's certificate.

Truststore Actions: This option allows the operator to create a trust keystore for the monitor and import certificates.

Credentials: This option allows the safe storage of the credentials required to communicate with the NMS cesejb, nms-ws, and emcli services.

In addition a list of outstanding issues that have still to be performed are listed. This is a subset of the options listed above. As options are completed they are removed from the outstanding issues.

As the NMS Monitor is configured, a property file (msmonitor.properties) is automatically created in the WebLogic $DOMAIN_HOME.

Configure ZooKeeper

Select the Configure ZooKeeper option to configure the ZooKeeper properties. The tool displays the current settings:

zookeeper.connectString: This is the connection string used to connect each node in the ZooKeeper cluster.

zookeeper.namespace: This is the ZooKeeper namespace. Default: nmsmonitor.

zookeeper.protocol: This is the protocol to use to communicate to the cluster. Valid values are http or https.

The following options are available:

1 : New connect string

2 : Add new node to connect string

3 : Set namespace

4 : Set protocol

New Connect String

Select the New connect string option to set the connection string for the cluster. For example to connect to the example ZooKeeper cluster with secure client ports configured on 2182:

```
server.1=<server1_hostname>
server.2=<server2_hostname>
server.3=<server3_hostname>
```

The connection string would be:

```
<server1_hostname>:2182, <server2_hostname>:2182,
<server3_hostname>:2182
```

Add New Node to Connect String

Select this option to add an additional ZooKeeper node to the existing connection string.

Set Namespace

Select this option to configure the ZooKeeper namespace if used.

Set Protocol

Select this option to configure whether the connection to the ZooKeeper node is secure (https) or not (http).

Configure Datasource

Select the **Configure datasource** option to configure which database datasources to use to communicate with the NMS database. The list of existing datasources can be found in the WebLogic administration console -> Services -> Datasources. The datasources used should connect to the read/write schema of the corresponding NMS environment. The tool displays the current settings:

config.datasource_1: This is the datasource to use for the NMS instance in environment 1.

config.datasource_2: This is the datasource to use for the NMS instance in environment 2.

The following options are available:

1 : Set datasource 1

2 : Set datasource 2

Set datasource 1

Select this option to configure the datasource to use for the environment 1 NMS instance's database schema.

Set datasource 2

Select this option to configure the datasource to use for the environment 2 NMS instance's database schema.

Configure emcli

Select the Configure emcli option to configure the location of the Enterprise Manager Command Line Interface. The tool displays the current settings:

emcli.home: The home directory where the emcli is installed.

emcli.state_dir: The emcli state directory if applicable. Sets the EMCLI_STATE_DIR environment variable when running emcli. This is typically not needed.

emcli.options: The emcli options if applicable. Sets the EMCLI_OPTS environment variable when running emcli. This is typically not needed.

emcli.header.primary_system: The Primary System header in the output for the "emcli get_operation_plans" command. Default: Primary System.

emcli.header.standby_system: The Standby System header in the output for the "emcli get_operation_plans" command. Default: Standby System.

emcli.header.plan_name: The Plan Name header in the output for the "emcli get_operation_plans" command. Default: Plan Name.

emcli.header.target_name: The Target Name header in the output for the "emcli get_targets" command. Default: Target Name.

The following options are available:

- Set home: Select this option to set the home directory of the emcli.
- Set homeSelect this option to set the state directory of the emcli.
- Select this option to set the options.
- Set header primary system
- Select this option to set the primary system header.
- Set header standby system
- Select this option to set standby system header.
- Set header plan name
- Select this option to set the plan name header.
- Set header target name
- Select this option to set the target name header.

Keystore Actions

Select the **Keystore actions** option to configure the identity keystore. The tool displays the current settings for the keystore:

credentials.keystore

This is the location of the identity store. The default location is $DOMAIN_HOME/security/nmsmon_keystore.p12

credentials.keystore_password

This is the encrypted password used to access the identity store.

The following options are available:

1 : Set keystore location

2 : Create keystore

3 : Export certificate

Set Keystore Location

Select the **Set keystore location** option to configure the keystore location. The operator will be prompted for the location of the keystore. Enter the location or hit return to select the default option.

If the file does not exist, the keystore actions page shall indicate that the file does not exist. The keystore is generated using option 2 Create keystore.

Create Keystore

Select the **Create keystore** option to create the identity keystore and save its password in encrypted form. The tool will prompt for the following information:

Enter Password:

Enter the desired password for accessing the new identity store.

Re-enter Password:

The operator is asked to confirm the password.

The tool will then continue to prompt for the Distinguished Name information for the NMS Agent certificate. An example is given below. Hitting return at each prompt will select the default value Unknown.

What is your first and last name? (Unknown) = nms@example.com

What is the name of your organizational unit? (Unknown) = GBU

What is the name of your organization? (Unknown) = Oracle

What is the name of your City or Locality? (Unknown) = Minneapolis

What is the name of your State or Province? (Unknown) = Minnesota

What is the two-letter country code for this unit? (Unknown) = US

The operator is then prompted to confirm the details.

Is CN=nms@example.com,OU=GBU,O=Oracle,L=Minneapolis,ST=Minnesota,ST=Minnesota correct? (no) =

Enter yes to confirm the details or hit return to select the default option no. Selecting no will cancel the action and the keystore shall not be created.

The operator is then prompted for the valid duration of the server certificate that will be created.

Enter validity period in days (365)

Enter a period in days or hit return to select 365 days.

The identity keystore is then created in the correct location and the encrypted password is set in the NMS Monitor property file.

Export certificate

The NMS Monitor certificate must be added to the truststore of each NMS Agents that communicates with the agent and each of the ZooKeeper nodes. In order to do this the certificate needs to be exported to file. Select the export certificate option to export the agent's certificate.

The operator shall be prompted for the keystore password.

After the password is correctly entered, the certificate is automatically exported to the file:

`nmsmonitor.cert`

Truststore Actions

Select the **Truststore action**s option to configure the truststore for the NMS Monitor. The tool displays the current settings for the truststore:

credentials.truststore

This is the location of the identity store. The default location is $DOMAIN_HOME/security/nmsmon_truststore.p12

credentials.truststore_password

This is the encrypted password used to access the truststore.

The following options are available:

1 : Set truststore location

2 : Create truststore

3 : Import certificate

Set Truststore Location

Select the **Set truststore location** option to configure the truststore location. The operator will be prompted for the location of the truststore. Enter the location or hit return to select the default option.

If the file does not exist, the truststore actions page shall indicate that the file does not exist. The truststore is generated using option 2 Create truststore.

Create Truststore

Select the **Create truststore** option to create the truststore and save its password in encrypted form. The tool will prompt for the following information:

Enter Password:

Enter the desired password for accessing the new identity store.

Re-enter Password:

The operator is asked to confirm the password.

The identity truststore is the created in the correct location and the encrypted password is set in the NMS Monitor property file.

Import Certificate

Select the **Import certificate** option to import a certificate into the monitor's truststore. Each NMS Agents and ZooKeeper nodes that communicates with the agent will require its certificate to be imported into the monitor's truststore.

The operator shall be prompted for the truststore password.

The operator shall then be prompted for the alias they wish to store the certificate under.

Enter alias =

The operator should enter the name of the alias for the certificate being imported.

The operator will then be asked for the location of the certificate on the filesystem.

Enter certificate location (myalias.cert) =

Enter the location of the certificate or hit return to use the default location of <alias>.cert

The certificate will then be loaded into the truststore.

Credentials

In order for the NMS Monitor to communicate to the cesejb & nms-ws (both instances), WebLogic admin and emcli services a set of credentials are required to be configured. This option allows the credentials to be viewed and managed. The tool displays the current settings:

- A list of currently stored credentials and their encrypted value.

The name of each credential takes the form:

```
<hostname>:<port>
```

Where hostname is the <hostname> of the server the service is installed on and <port> is that services listen port.

The following credentials must be configured in the NMS Monitor:

emcli

Credentials for logging into the Enterprise Manager command line interface.

Hostname=emcli

Port=0

Username=<admin account username>

Password=<admin account password>

Cesejb, nms-ws

Credentials for using the WebLogic cesejb and nms-ws deployments. This needs to be done for each of the WebLogic servers deploying these services.

Hostname=<hostname of the WebLogic server>

Port=<the secure port number for the WebLogic server>

Username=<valid NMS username>

Password=<valid NMS password>

Application=<valid mobile application id>

WebLogic Admin

Credentials for using the WebLogic Admin must be provided.

Hostname=<hostname of the WebLogic server>

Port=7001

Username=<the WebLogic admin username>

Password=<the WebLogic admin password>

The following options are available:

1 : Create new credential

2 : Delete credential

Create New Credential

Select this option to create a new credential. You will be prompted for the following information:

Hostname

Enter the hostname of the service.

Port

Enter the listen port of the service.

Username

Enter the username to login with

Enter password

Enter the password to login with.

Re-enter password

Revalidate the password.

Application.

This is the mobile application id and is applicable only to the nms-ws service. For all other services simply hit return.

Delete Credential

Select this option to delete one of the listed credentials. You will be prompted for the credentials hostname and port.

Creating the NMS Monitor Server

The NMS Monitor needs to be deployed to a new managed server in the WebLogic instance. For full details on how to create a managed service refer to the Installation Guide -> System Installation -> WebLogic Server Runtime Configuration.

Create the Server

Invoke the WebLogic admin console using the following URL

```
http://<hostname>:<port>/console
```

Here hostname represents the DNS name or IP address of the Administration Server, and port represents the number of the port on which the Administration Server is listening for requests (port 7001 by default).

Navigate to Environment -> Servers

Select New to create a new server

Enter the server name and give it a unique listening port. Create the server as “Stand Alone”.

Select the server in the server list to edit its details. This will invoke the server details page for the new server.

Enter the hostname of the server as the listen address.

Enable the SSL listen port and set the port to a unique port number.

Save the changes.

Edit the advanced settings and set and set the RMI JDBC Security to Secure.

Edit the keystore options and make sure the default java truststore is configured.

Save the changes.

Configure SSL

In order to enable SSL for the NMS Monitor web application the keystores need to be configured.

Navigate to Environment -> Servers

Select the NMS Monitor server

Select the Keystores tab

Select the change/new button

Select the “Custom Identity and Java Standard Trust” menu option

Save the selection.

In the identity section:

In the Custom Identity Keystore field enter the location of the NMS Monitor keystore, e.g.

/scratch/gbuora/Oracle/Middleware/Oracle_Home/user_projects/domains/nms/security/nmsmon_keystore.p12

In the Customer Identity Keystore Type field leave as the default option JKS

Enter and confirm the NMS Monitor keystore password.

Save the changes.

Select the SSL tab.

In the Private Key Alias field enter the alias of the NMS Monitor private key ‘nmsmonitor’

Enter and confirm the NMS Monitor keystore password.

Save the changes.

Starting the Server

Invoke the WebLogic admin console and navigate to Environment->Servers

Select the **Control** tab.

Select the tick box beside the NMS Monitor server, then select the Start command.

Update the JDBC Datasources

The JDBC datasources for the NMS database need to be accessible to the new servers.

In the WebLogic admin console select Services, Data Sources

Select each of the NMS data sources in turn and perform the following actions:

Select the Targets tab

In the targets list, select the newly created NMS Monitor server in addition to the targets already selected.

Save the datasources.

Deploying the NMS Monitor

The new NMS Monitor service needs to be deployed to the new NMS Monitor server in the WebLogic instance.

Copy $NMS_BASE/dist/install/nms-monitor.ear from the NMS installation to the WebLogic server.

Invoke the WebLogic admin console using the following URL

```
http://<hostname>:<port>/console
```

Here hostname represents the DNS name or IP address of the Administration Server, and port represents the number of the port on which the Administration Server is listening for requests (port 7001 by default).

Navigate to Deployments -> Configuration

Select the Install button

Navigate to the path where NMS Monitor is installed and select the nms-monitor.ear

Select next

Select Install this deployment as an application

Select next

In the list of available targets, select the new NMS Monitor server

Select next, next, finish to complete the deployment.

NMS CESEJB Self-Signed Certificates

If the NMS CESEJB has been self-signed the certificate must be imported into the default java truststore in order for the NMS Monitor to trust it.

The certificate can be obtained from the NMS keystore or exported from a browser, e.g. Navigate to

Where <hostname> is the hostname of the WebLogic server and <port> is the secure port used for connecting to the NMS Applications.

View and export the certificate chain.

The certificate chain can then be imported into the default java truststore using the following command:

keytool -import -alias <alias> -file <certificate> -keystore $JAVA_HOME/lib/security/cacerts

NMS Agent

NMS Agent Overview

The NMS Agent runs on each NMS Services server and monitors NMS back-end services.

The NMS Agent is a Java process which runs as the NMS administrative user on the machine where NMS Services run. With dual-environment configuration, both administrative users have an NMS Agent instance installed and running. The Agent has various monitor tasks, which store the most recent results in memory.

Status Monitor

The status monitor runs /usr/bin/env in a ksh login shell to get the current NMS environment variables for the administrator user. It is therefore a requirement that the $HOME/.profile file of the administrator user sources the .nmsrc script. The status monitor periodically executes the NMS utility oem-util in this environment to retrieve the current state of NMS.

File Monitor

The file monitor monitors the following files for updates and upon detection of changes will initiate a reload of the Status Monitors environment:

```
$HOME/.nmsrc
$NMS_HOME/.nms-site-rc
```

In addition the following file is monitored in order to maintain the current NMS version installed on the server:

```
$NMS_ROOT/nms/inventory/nms_version.xml
```

Web Service

The NMS agent listens for https requests with the following end point:

GET /nmsagent/v1/status

This is used by the NMS Monitor to periodically return the current and previous state of the services, timestamps and the NMS version.

Security

The NMS Agent web service will requires two-way authentication. See NMS Agent for full details on configuration.

Logs

NMS Agent logs to the $NMS_LOG_DIR/nmsagent.log. Daily log rotation places the old logs into the $NMS_LOG_DIR/old_logs/ directory.

Configuration

The NMS Agent is configured via the nmsagent.properties configuration file located in the administrator users `$NMS_HOME/etc` directory. A configuration tool, nms-agent-config is provided to help in the configuration of the NMS Agent.

The NMS Agent should be configured to run as a service for each administrator user on the servers where the NMS Services run. With dual-environment configuration, the NMS Agent must be configured on both administration users. For this sake of this document these are assumed to be nms_1 and nms_2. One NMS Agent should be configured per user.

Administration User Profile

It is a requirement that the `$HOME/.profile` file of the administrator user sources the `.nmsrc` script.

Configuration Tool Main Page

To configure an NMS Agent on a server run the interactive nms-agent-config utility. This utility must be executed on both NMS administration users.

The main page displays the following options:

1 : Display properties

2 : Configure server

3 : Keystore actions

4 : Truststore actions

5 : Generate service script

0 : Exit

Display properties: Selecting this option displays the current configuration properties for the NMS Agent.

Configure server: This option allows the server properties to be configured including hostname and port.

Keystore actions: This options allows the operator to create an identity keystore for the agent and export the agent's certificate.

Truststore actions: This option allows the operator to create a trust keystore for the agent and import certificates.

Generate service script: This option provides the configuration and commands necessary to run the NMS Agent as a service.

In addition a list of outstanding issues that have still to be performed are listed. This is a subset of the options listed above. As options are completed they are removed from the outstanding issues.

As the NMS Agent is configured a property file /etc/nmsagent.properties is automatically created in the user's home directory to store the configuration.

Configure Server

Select the Configure server option to configure the server properties. The tool displays the current settings for the server:

server.host

This is the hostname of the server running the NMS Services

server.port

This is the listen port that the NMS Agent shall use, default is 9988.

server.protocol

This is the protocol used to communicate to the NMS Monitor, valid values are http or https

nmsstatus.interval

This is the timout interval used to determine how often the NMS Agent checks the status of the NMS Services. This parameter is in milliseconds and defaults to 5000.

The following options are available:

1 : Set hostname

2 : Set port

3 : Set protocol

4 : Set interval

Set Hostname

Select the Set hostname option to set the hostname of the server. Enter the hostname of the server.

Set Port

Select the Set port option to set the listen port. Enter the port number or hit return to use the default value, 9988. Make sure to choose a unique port for each user instance on the system.

Set Protocol

Select the **Set protocol** option to set the transport protocol. Enter https for secure protocol or hit return to default to http protocol.

Set Interval

Select the **Set interval** option to set the timeout interval. Enter the interval or hit return to default to 5000 milliseconds.

Keystore Actions

Select the `Keystore actions` option to configure the identity keystore. The tool displays the current settings for the keystore:

credentials.keystore

This is the location of the identity store. The default location is etc/nmsagent_keystore.p12 in the user's home directory.

credentials.keystore_password

This is the encrypted password used to access the identity store.

The following options are available:

1 : Set keystore location

2 : Create keystore

3 : Export certificate

Set Keystore Location

Select the **Set keystore location** option to configure the keystore location. The operator will be prompted for the location of the keystore. Enter the location or hit return to select the default option.

If the file does not exist, the keystore actions page shall indicate that the file does not exist. The keystore is generated using option 2 Create keystore.

Create Keystore

Select the **Create keystore** option to create the identity keystore and save its password in encrypted form. The tool will prompt for the following information:

Enter Password:

Enter the desired password for accessing the new identity store.

Re-enter Password:

The operator is asked to confirm the password.

The tool will then continue to prompt for the Distinguished Name information for the NMS Agent certificate. An example is given below. Hitting return at each prompt will select the default value Unknown.

What is your first and last name? (Unknown) = nms@example.com

What is the name of your organizational unit? (Unknown) = GBU

What is the name of your organization? (Unknown) = Oracle

What is the name of your City or Locality? (Unknown) = Minneapolis

What is the name of your State or Province? (Unknown) = Minnesota

What is the two-letter country code for this unit? (Unknown) = US

The operator is then prompted to confirm the details.

Is CN=nms@example.com,OU=GBU,O=Oracle,L=Minneapolis,ST=Minnesota,ST=Minnesota correct? (no) =

Enter yes to confirm the details or hit return to select the default option no. Selecting no will cancel the action and the keystore shall not be created.

The operator is then prompted for the valid duration of the server certificate that will be created.

Enter validity period in days (365)

Enter a period in days or hit return to select 365 days.

The identity keystore is the created in the correct location and the encrypted password is set in the NMS Agent property file.

Export Certificate

The NMS Agents certificate must be added to the truststore of each NMS Monitor that communicates with the agent. In order to do this the certificate needs to be exported to file. Select the export certificate option to export the agent's certificate.

The operator shall be prompted for the keystore password.

After the password is correctly entered, the certificate is automatically exported to the file:

`nmsagent-<hostname>-<username>.cert`

Truststore Actions

Select the **Truststore actions** option to configure the truststore for the NMS Agent. The tool displays the current settings for the truststore:

credentials.truststore

This is the location of the identity store. The default location is etc/ nmsagent_truststore.p12 in the user's home directory.

credentials.truststore_password

This is the encrypted password used to access the truststore.

The following options are available:

1 : Set truststore location

2 : Create truststore

3 : Import certificate

Set truststore location

Select the **Set truststore** location option to configure the truststore location. The operator will be prompted for the location of the truststore. Enter the location or hit return to select the default option.

If the file does not exist, the truststore actions page shall indicate that the file does not exist. The truststore is generated using option 2 Create truststore.

Create truststore

Select the Create truststore option to create the truststore and save its password in encrypted form. The tool will prompt for the following information:

Enter Password:

Enter the desired password for accessing the new identity store.

Re-enter Password:

The operator is asked to confirm the password.

The identity truststore is the created in the correct location and the encrypted password is set in the NMS Agent property file.

Import Certificate

Select the **Import certificate** option to import a certificate into the agent's truststore. Each NMS Monitor that communicates with the agent will require its certificate to be imported into the agent's truststore.

The operator shall be prompted for the truststore password.

The operator shall then be prompted for the alias they wish to store the certificate under.

Enter alias =

The operator should enter the name of the alias for the NMS Monitor certificate being imported.

The operator will then be asked for the location of the certificate on the filesystem.

Enter certificate location (myalias.cert) =

Enter the location of the certificate or hit return to use the default location of <alais>.cert

The certificate will then be loaded into the truststore.

Generate Service Script

To allow the NMS Agent to execute on server start-up it needs to be configured as a service. Selecting the Generate service script option will generate the necessary configuration to run the agent as a service and show the operator the commands that need to be executed.

The tool displays the set of commands required to create a service once the service script has been generated.

Select option 1 to generate the service file

then use the following commands to configure the service:

```
- sudo cp nmsagent-<username>.service /usr/lib/systemd/system/
- sudo systemctl enable nmsagent-<username>
```

Use the following commands to manage the service:

```
- sudo systemctl start nmsagent-<username>
```

```
- sudo systemctl stop nmsagent-<username>
- sudo systemctl status nmsagent-<username>
```

The following options are available:

1 : Generate Service Script

Generate Service Script

Select **Generate Service Script** to create the service script used to configure the service.

The operator shall be prompted for the user group they wish the service to run under. Enter a user group that the current user belongs to.

The script is then generated with the following file format:

```
nmsagent-<username>.service
```

The format of the file is as follows:

[Unit]

Description=NMS Agent <username>

ConditionFileIsExecutable=/home/<username>/nmsagent/bin/nms-agent

Requires=network-online.target local-fs.target

After=network-online.target local-fs.target

[Service]

User=<username>

Group=<usergroup>

Type=forking

PIDFile=/home/=<username>/logs/nmsagent.pid

ExecStart=/home/=<username>/nmsagent/bin/nms-agent start

ExecStop=/home/=<username>/nmsagent/bin/nms-agent stop

Restart=on-failure

RestartSec=5

Environment="NMS_HOME=/home/<username>"

Environment="NMS_BASE=/home/<username>"

Environment="NMS_LOG_DIR=/home/<username>/logs"

[Install]

WantedBy=multi-user.target

The nmsagent Directory

The NMS Agent is normally installed under $NMS_BASE/nmsagent

This must be copied to $NMS_HOME/nmsagent each time a new version of nmsagent is installed. This will avoid the need to regenerate the service script each time a new release is installed. This can be done with the following command: rsync -avL --delete "$NMS_BASE/nmsagent/" "$NMS_HOME/nmsagent"

It is important to make sure that the executable to $NMS_HOME/nmsagent/bin/nms-agent is a valid link or executable.

NMS Monitor Web Application

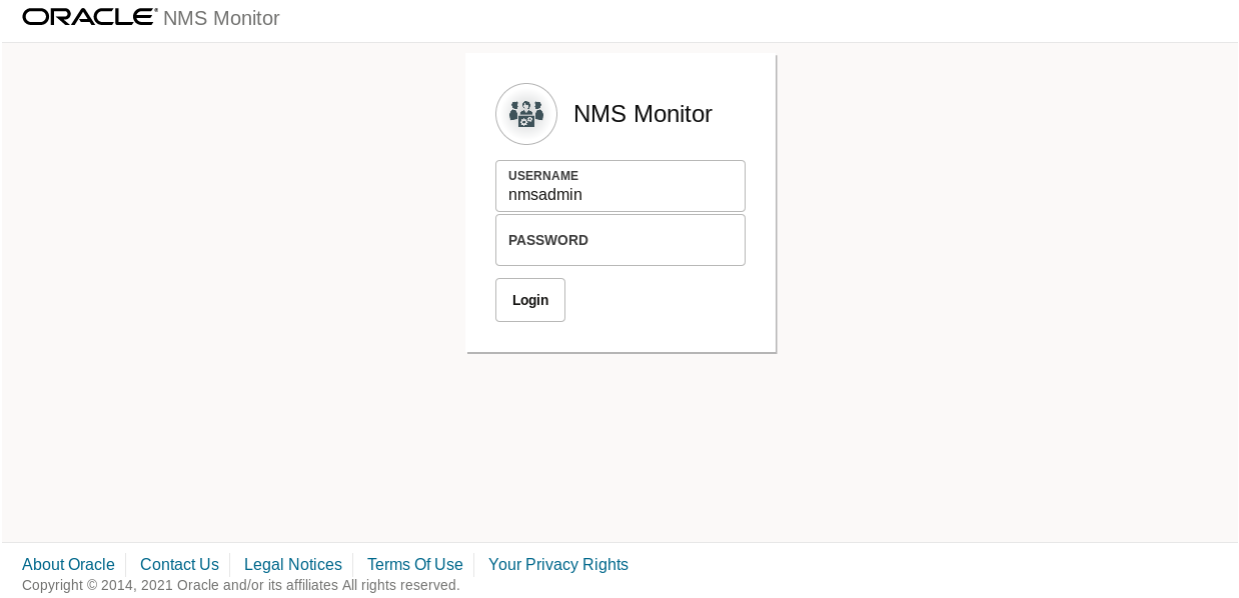
NMS Monitor application allows users to:

1. Monitor status of NMS installation.
2. Enter configuration parameters for monitoring NMS status and initiating automatic failover
3. Manually initiate failover

Launching NMS Monitor and Logging In

NMS Monitor is a web application that is accessed through web browser by opening an URL (Monitor Server>:<port>/NmsMonitor/).

Upon accessing the application URL you will be presented with the login screen.



To log in, enter the following information on the login screen:

USERNAME: enter the user name provided to you by your system administrator.

PASSWORD: enter the password provided to you by your system administrator.

Login: click the **Login** button to verify your login credentials.

If correct login credentials are entered the application will proceed to the main screen. The user must be in the nms-admin or nmsuser group. Only users in the nms-admin group will be allowed to save configuration changes or initiate a manual failover.

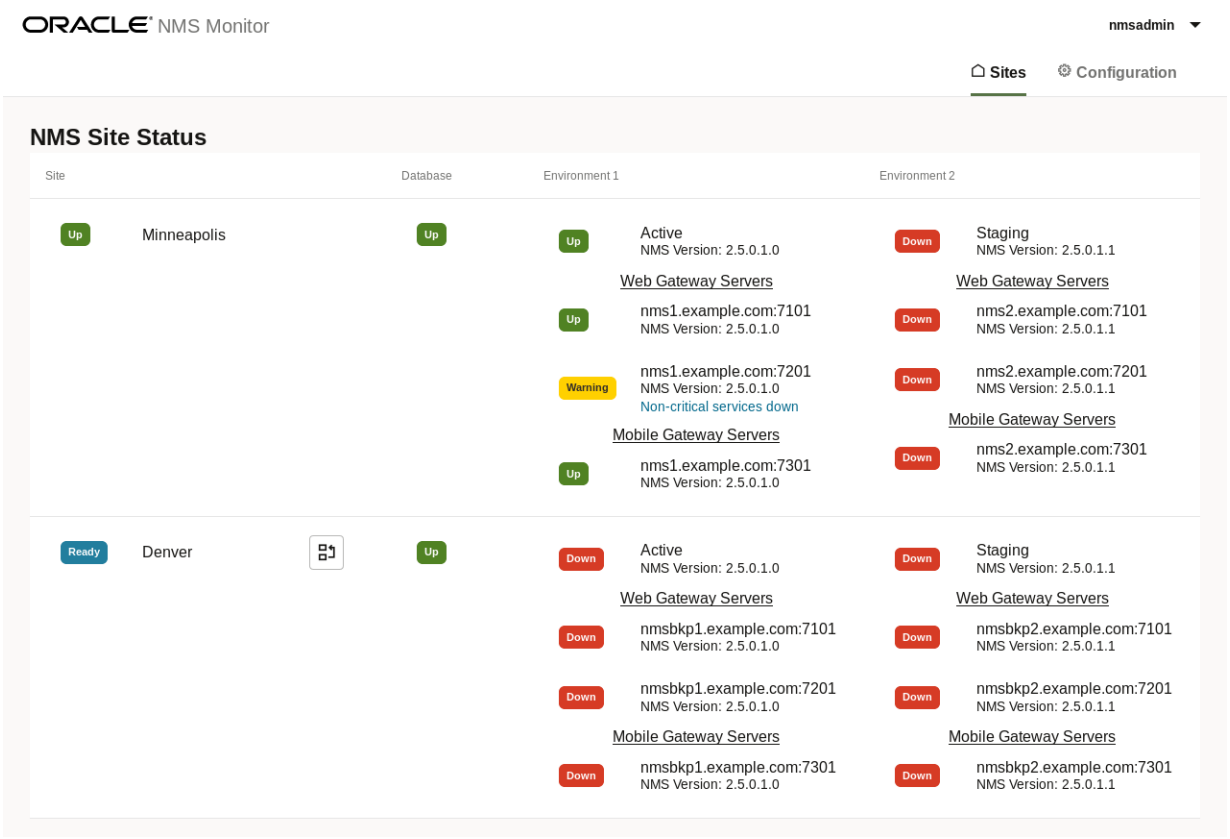
NMS Monitor Main Screen

Main screen of the application consists of the following elements:

- Application Title.
- User Menu: displays username and allows you to sign out from the application.
- Navigation Bar: allows you to navigate to different application screen (Sites and Configuration).
- Content Area.

NMS Site Status Screen

NMS Site Status is the screen displayed after initial login into the application. It can also be accessed through the Sites option in the Navigation Bar.



NMS Site Status screen displays the current status of the monitored NMS installation. Information is presented as a table where each row corresponds to NMS site (for example, primary and backup datacenters).

The table has following columns:

Column	Description
Site	Site name, overall site status, and button to initiate failover to this site manually
Database	Database status
Environment 1	Status of the NMS environment 1
Environment 2	Status of the NMS environment 2 (if present)

Status of NMS environment includes:

- Overall environment status
- Environment type (Action or Staging)
- Installed NMS version
- Status of Web Gateway servers
- Status of Mobile Gateway servers

For each server status, hostname and port number, and installed NMS version are displayed.

Possible Status Values

- Up – component is fully operational
- Warning – component is operational with warning messages
- Down – component is not operational
- Error – Error retrieving status
- Unknown – status information unavailable

The following statuses are only applicable to standby sites.

- Ready – site is ready to be failed over to
- Not Ready – site is not ready to be failed over to

Each item in the NMS Site Status table can have zero or more associated status messages. If present, the first message is displayed and the remaining messages can be viewed by clicking on the first message (it is displayed as a link) or by hovering over it.

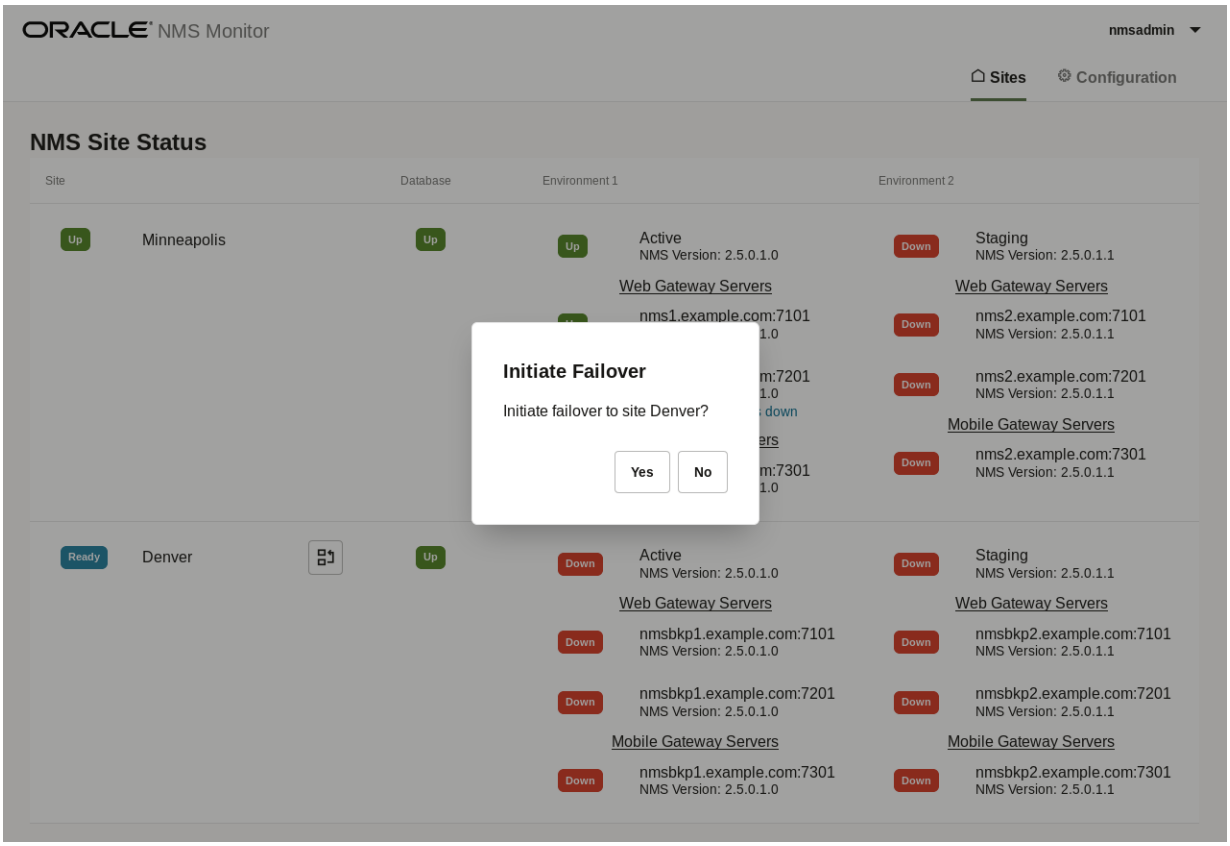
Information on the NMS Site Status screen is automatically updated every 30 seconds.

Manual Failover

For NMS sites which are ready for failover (site status is 'Ready') user can manually initiate failover process.

To initiate manual failover:

1. Verify that the desired failover target site has Ready status.
2. Click the **Initiate failover** button for the site.
3. Answer Yes in the confirmation dialog box to initiate failover process.



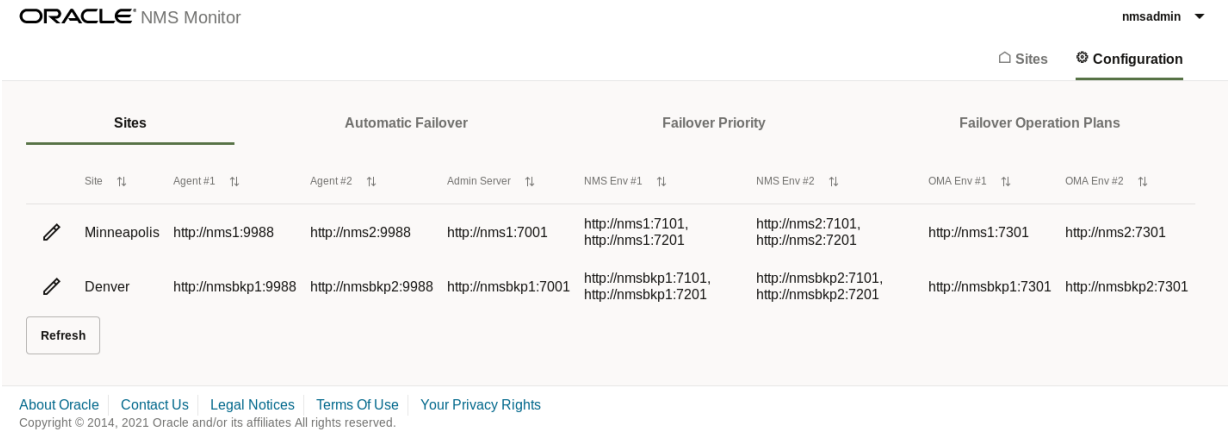
Configuration Screen

Configuration screen allows you to specify parameters for monitoring NMS installation and initiating automatic failover. It can also be accessed through the Configuration option in the Navigation Bar. The configuration screen contains several tabs.

Information on each tab of the Configuration screen can be refreshed by pressing **Refresh** button, which is present on each tab. Refreshing information will discard any unsaved changes.

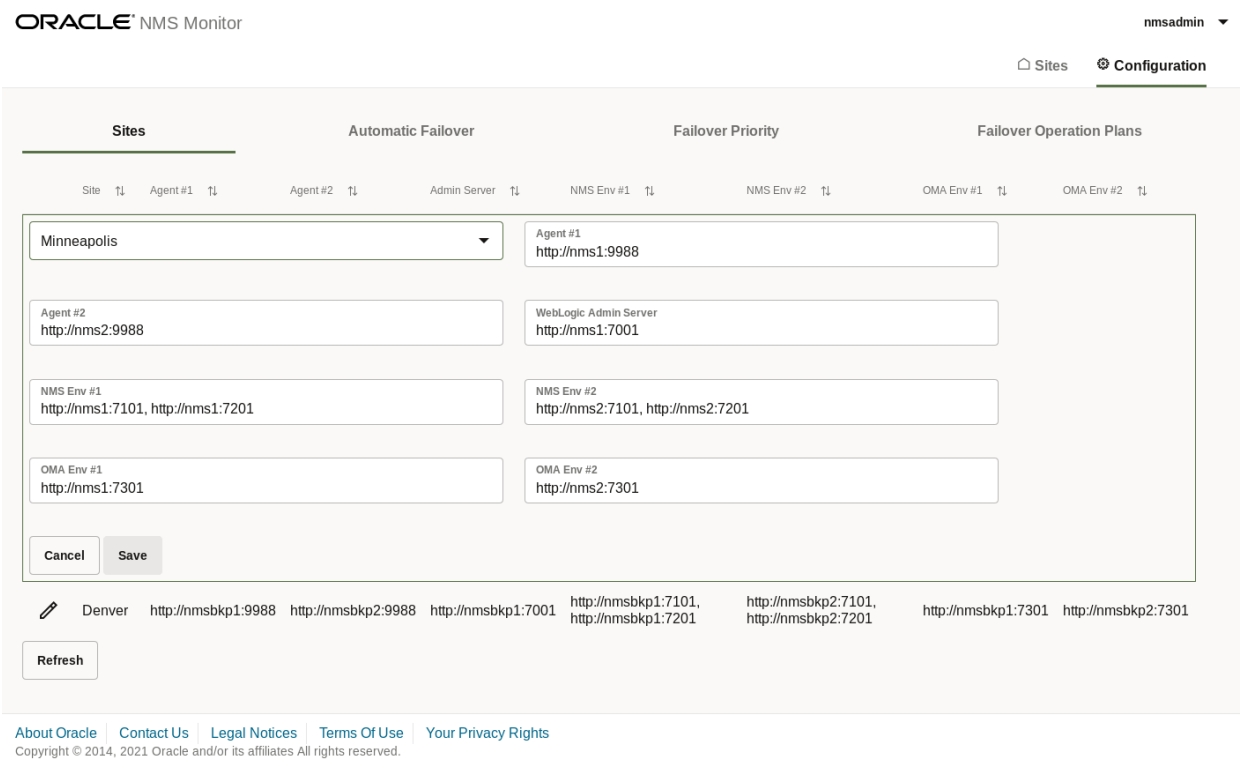
Sites

The **Sites** tab allows to view and edit connection information for the monitored NMS components. Information is displayed as a table with one row per site.



To edit information:

1. Click the **Edit** button (pencil icon).
2. Edit information in the presented form.
3. Click the **Save** button to save changes or Cancel button to discard them.



Connection information for the following components can be viewed/edited:

- Site Name – name of the site. This must match the name of the site in Site Guard
- Agent #1 – monitoring agent for environment 1
- Agent #2 – monitoring for environment 2
- WebLogic Admin Server – WebLogic Admin server
- NMS Env #1 – WebLogic servers supporting NMS users in environment 1
- NMS Env #2 – WebLogic servers supporting NMS users in environment 2
- OMA Env #1 - WebLogic servers supporting OMA users in environment 1
- OMA Env #2 - WebLogic servers supporting OMA users in environment 2

Connection information is specified as an URL in the format

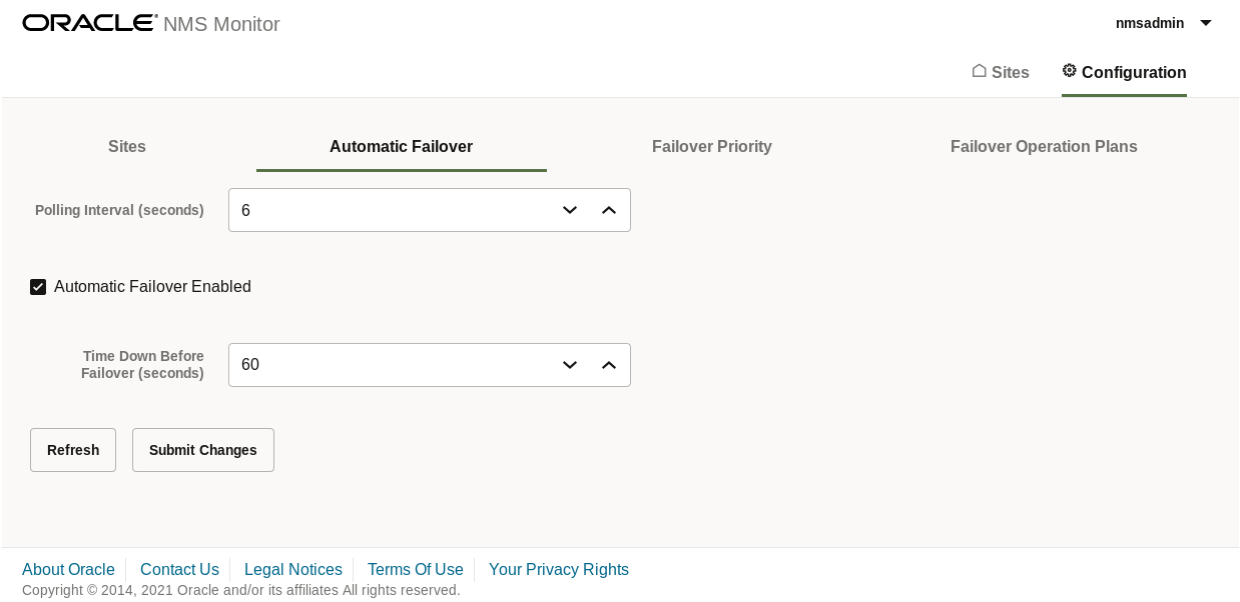
```
<protocol>://<hostname or IP address>:<port number>
```

Allowed protocols are: http, https, t3, t3s.

For NMS and OMA WebLogic servers comma-separated list of URLs can be entered. The hostname must match the Listen Address configured in the corresponding WebLogic servers.

Automatic Failover

The Automatic Failover tab contains parameters that control automatic failover between NMS sites.



Polling Interval is the number of seconds between subsequent checks of the statuses of monitored components.

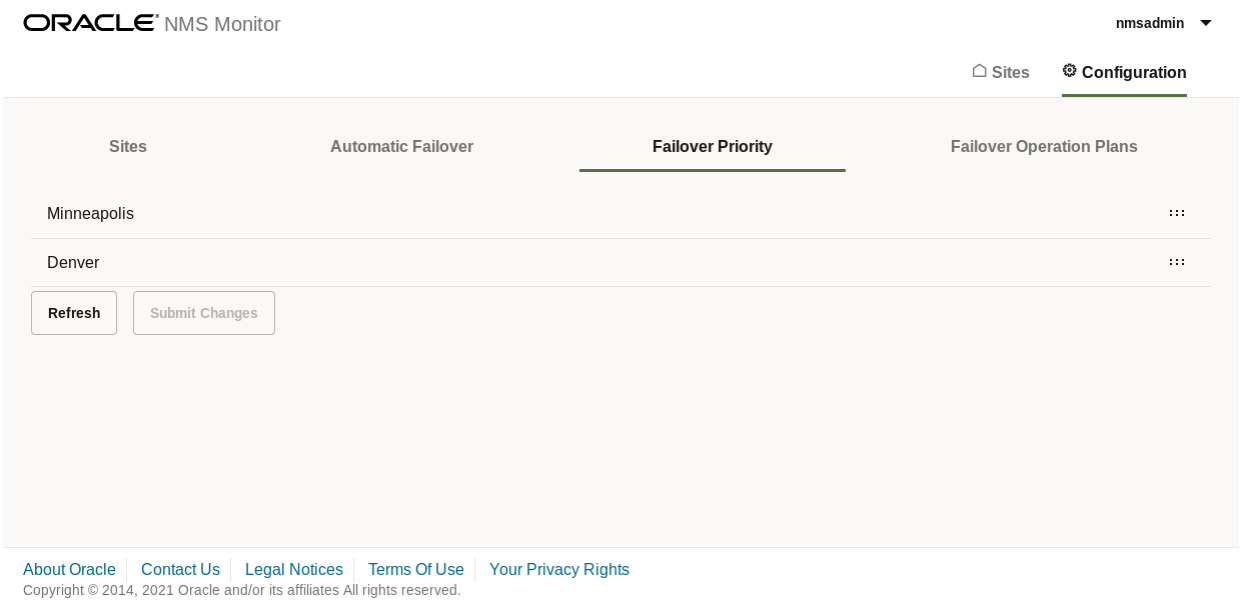
Automatic Failover Enabled checkbox allows to enable or disable automatic failover.

Time Down Before Failover is the number of seconds between the moment when failure of NMS site is detected and start of automatic failover process.

Press **Submit Changes** button to save modification made on this tab.

Failover Priority

Failover Priority tab allows to specify relative priority of NMS sites for the automatic failover process. Sites that are above in the list have higher priority.

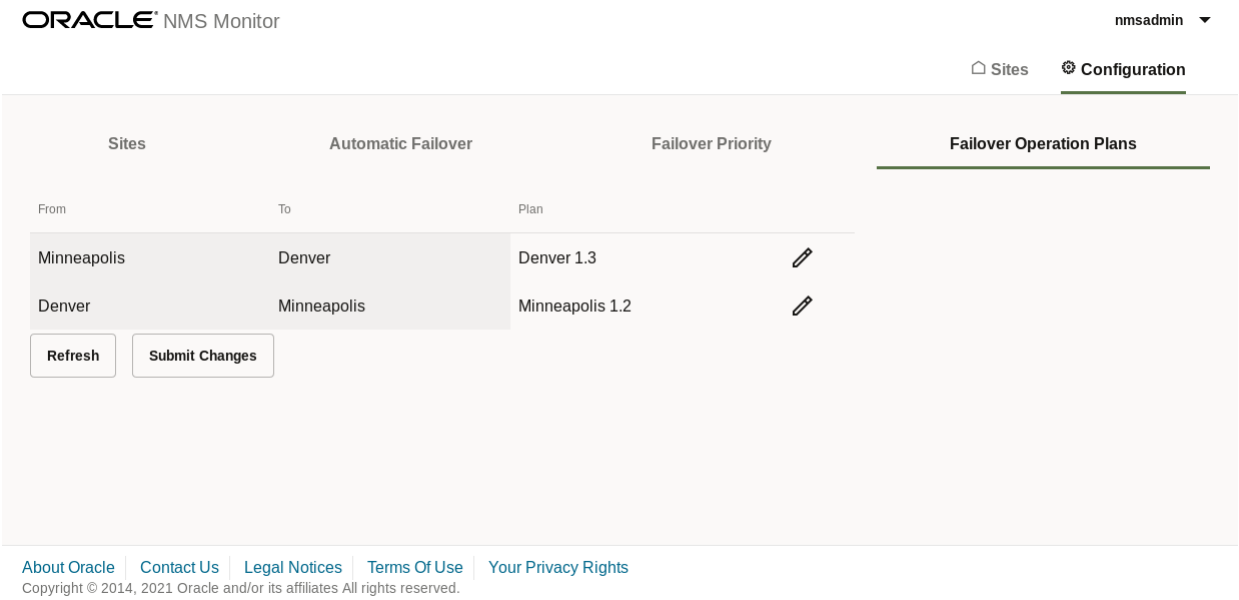


To modify site failover priority:

1. Press and hold left mouse button on the “drag handle” icon.
2. Drag and drop site row into the desired position.
3. Click the **Submit Changes** button to save modification made on this tab.

Failover Operation Plans

Failover Operation Plans tab allows you to select failover plan to be used to failover between each pair of NMS sites.



- From –NMS site to failover from
- To – NMS site to failover to
- Plan – failover plan to use

To specify failover plan:

1. Click the **Edit** button or double-click in the Plan column to enter edit mode.
1. Select the plan from the provided list of options.

Click the **Submit Changes** button to save modification made on this tab. These plans must be configured before manual or automatic failover will work.

Chapter 12

Troubleshooting and Support

If you experience problems with your Oracle Utilities Network Management System, there are a number of tools and resources available to help you identify and resolve problems. These include log files, core files, Knowledge Management Documents available on My Oracle Support, and Oracle Customer Support.

This chapter includes the following topics:

- Troubleshooting an Issue
  - Evaluating System Status
  - Examining Log Files
  - Examining Core Files
  - Identifying Memory Leaks with monitor-ps-sizes
  - Other Troubleshooting Utilities
- Oracle Support Information
  - Support Knowledgebase
  - Contacting Oracle Support

Troubleshooting an Issue

Evaluating System Status

A good first diagnosis is to run the Unix **top** command or equivalent (**topas** on AIX; **prstat** on Solaris). This will display information such as what processes are running, current memory usage, and free memory.

Examining Log Files

Log files are the best tools for tracking down the source of a problem because very seldom does something crash or behave strangely without an entry being logged. Before reporting an issue to Oracle Customer Support, it is important to review log files for critical information that may help Oracle Customer Support solve your problem.

There are several logs that are especially useful for troubleshooting issues with NMS implementations; these include services logs and PID logs. The sections that follow describe important troubleshooting logs.

Oracle Utilities Network Management System Log Files

Application log files are located in the directory specified by the **NMS_LOG_DIR** environment variable, which is defined in the `.nmsrc` file.

Note: by default, **NMS_LOG_DIR** is set as `$NMS_HOME/logs`

- There will be one log file in this directory for each actively running service.
- After a process has been stopped and restarted, the old log file for that particular server is moved to the `old_log` subdirectory within the `NMS_LOG_DIR` directory.
- After the number of days specified in `$NMS_DAYS_TO_LOG`, old log files for a given process in the `$NMS_LOG_DIR/old_log` directory will be purged on the next attempt to start that process. The default for `NMS_DAYS_TO_LOG` is 7 (days). Thus, old logs will only be retained for 1 week by default.

Service Logs

Looking for DBService errors is a common starting place in determining if the problem is a database issue or a services issue. DBService errors can appear in DBService, TCDBService, and MBDBService or some other *DBService, depending upon which service is having a problem interacting with the database.

If a particular service cores, Customer Support will want to know if the service has any error messages in the log file right before it failed. The most relevant portion of the log is the text concerning what happened right before the dump. Often, there are important messages explaining why the service exited.

Another key service log is the SMService log. This log records if/when SMService attempts to restart other services.

Oracle Utilities Network Management System Log File Naming Conventions

Within the log directory, the following naming conventions apply:

- There is one log file for each Service actively executing on the server. Service logs are named `[Service Name].[date].[time].log`. Example log files would be:

```
DBService.2010052898.111721.log
DDService.20100528.111800.log
```

Trimming and Archiving Application Oracle Utilities Network Management System Log Files

As log files grow, they generally need to be removed or archived. When determining the maximum size and content of log files, consider your company's needs:

- If accounting files need to be kept for an audit, a larger log file is justifiable. Backups of those files might even be in order.
- After the number of days specified in $NMS_DAYS_TO_LOG (environment variable), the old log files for a given process in the $NMS_LOG_DIR/old_log directory will be purged on the next attempt to start that process. The default for $NMS_DAYS_TO_LOG is 7 (days). Thus, old logs will only be retained for 1 week by default.

Issues like these should be carefully assessed, and you should develop a policy around your company's specific needs.

PID Logs

PID logs are files with an integer value suffixed by `.log`. PID logs are generated in one of two ways.

- **cmd snapshot command**. This will create PID logs for all Isis processes currently running, whether they are services or tools. They appear in the following locations:
  - Services will appear in the `$NMS_LOG_DIR/run.<service>` directory of the user that starts services.
  - Tools will appear in the directory where ceslogin was started (typically the HOME directory of the user). If a tool is started from the command line, it will appear in the directory where the tool was started.
- **kill -usr2 <pid>**. This will not kill the tool. Rather, it will send a signal to the process that will create a `[pid].log` for that PID.

  Note: You can do this multiple times, and the logs will append additional dumps into the same log file as long as the process continues to run. It will not remove or replace logs upon additional snapshots of the same process. Customer Service recommends that these logs be cleaned up upon the end of investigating an issue.

Java Application Server Log Files

The WebLogic server log files are written to the following location:

- `MW_HOME/user_projects/domains/DOMAIN_NAME/servers/SERVER_NAME/logs`

  where:

  - **MW_HOME:** Oracle WebLogic Server installation directory.
  - **DOMAIN_NAME:** WebLogic domain name used for Oracle Utilities Network Management System.
  - **SERVER_NAME:** WebLogic server name used for Oracle Utilities Network Management System.

Retrieving EJB Timing Logs and Statistics from WebLogic

There are two options for retrieving EJB timing statistics. The first keeps track of the count, max, min, total, and average response times. It logs the usage in 5 minute intervals. It puts a minimal load on the system, so it can be left on at all times, if desired.

The following is an example of the information returned (times are in milliseconds):

```
First timestamp: Mon Nov 14 16:04:13 CST 2011
Last timestamp: Mon Nov 14 17:59:13 CST 2011

     Sum    Count   Avg   Min    Max  Method
 1006264    14465    69     0   7072  PublisherBean.messageHandlerLocal
 1005608    14465    69     0   7072  PublisherBean.messageHandler
 1004780    14465    69     0   7072
PublisherBean.internalMessageHandler
  992921      380  2612     0   7010  ViewerBean.flmUpdateEventHandler
   21275   147067     0     0      9
PublisherBean.getLocalPublishedMessages
   19849   208070     0     0      9  PublisherBean.getPublishedMessages
   12139       24   505   505    505  ViewerBean.getLayers
   11005      574    19     3    143  Session.executeSQLQuery
    8439       72   117     0    351  ViewerBean.getInheritanceMapping
    5006       24   208   208    208  CrewOperations.getFieldLengths
    4954       39   127     3    222  Session.getAlias
```

```
    4952       39   126     3    222  Session.getAliases
    4669       24   194   194    194  ViewerBean.getPartitions
    4457   208822     0     0      0  PublisherBean.isOnline
    2982       24   124   124    124  SwmanServiceImpl.init
...
```

The second option enables logging each EJB call with a timestamp. Since this method can generate a significant amount of logging information, it should not be enabled unless needed. Here is an example of the log:

```
2011-11-14 16:05:44 nms1 PublisherBean.getLocalPublishedMessages 0
2011-11-14 16:05:45 nms1 PublisherBean.isOnline 0
2011-11-14 16:05:45 nms1 PublisherBean.getPublishedMessages 0
2011-11-14 16:05:45 nms1 PublisherBean.getLocalPublishedMessages 0
2011-11-14 16:05:45 nms1 PublisherBean.isOnline 0
2011-11-14 16:05:45 nms1 PublisherBean.getPublishedMessages 0
...
```

Enabling Statistics

To enable statistics, cd to the WebLogic domain bin directory and run the following (replacing the username, password, url and server name); the second line should be entered all on the same line:

```
. ./setDomainEnv.sh

java weblogic.WLST scripts/configure-statistics [username] [password]
weblogic1 t3://localhost:7001 [server_name]
```

There should not be any errors. You should see output that ends with the following:

```
Activation completed
EJB Statistics have been successfully enabled.

Exiting WebLogic Scripting Tool.
```

If there are errors, recheck the server name and connection information.

Enabling EJB Logging

To log each ejb call, add or uncomment out the line in $NMS_CONFIG/jconfig/build.properties:

```
weblogic.ejb_logging = true
```

Then install by running:

```
nms-install-config --java
```

Accessing Statistics

The statistics are accessed from the nms server (not necessarily the WebLogic server).

To access the stats, use the following script (replacing the user name, password, and url with your system):

```
wls-stats-summary [username] [password] http://localhost:7001 2019-11-
01 13:00 2019-11-02 13:00
```

The date/times are the start and end times that you are interested in. If only one date/time is entered, it will go to the end of the log. If no times are given, than the entire log will be transferred. The date may be omitted to get the information from the current day.

Accessing the Log

To access the logs, use this script, which takes the same parameters as above

```
wls-stats-dump [username] [password] http://localhost:7001 2019-11-01
13:00 2019-11-02 13:00
```

EJB_STATISTICS

As an alternative to EJB Logging, EJB_STATATISTICS can be enabled, which is similar to other NMS logging.

To enable EJB_STATISTICS logging, which will log each ejb method calll, run:

```
Action any.publisher* ejb debug EJB_STATISTICS=1
```

Alternatively, EJB_STATISTICS can be added to log4j.properites.

Configuring the Retention of Statistics and Logs

This is managed by WebLogic. It can be set up to either keep a certain size of the archive, or to keep a certain number of hours of information. The statistics information is stored in the HarvestedDataArchive, and the logs are stored in the EventsDataArchive. See the Weblogic documentation for more information:

https://docs.oracle.com/cd/E11035_01/wls100/wldf_configuring/config_diag_archives.html#wp1069508

Java Client Application Logs

Java client applications generate a log file each time the application is started. The log files are named according to the following convention:

```
[server name]_[application]_[server date]_[server time].log
```

For example, `xyz.opal.com_WebWorkspace_20200603_1106899.log`.

The log files are saved to the following locations:

Operating System	Path
Microsoft Windows 7/10	`C:\Users\[user]\AppData\Local\Temp\OracleNMS`
Linux	`/tmp/[user]/OracleNMS/`

In addition, the Java Web Start applications log to the Java console. If you want to see the log messages in real time, enable the Java console.

For example, in Microsoft Windows, enable the console by doing the following:

1. Open the Windows Control Panel.
2. Open the Java Control Panel by double-clicking on the **Java** icon.
3. Select the **Advanced** tab.
4. Set Java console parameter to ‘Show console’ (Java console will be started maximized) or ‘Hide console’ (Java console will be started minimized).

Error Codes

NMS utilizes error codes to help implementers and support personnel diagnose issues with the system. Error codes are not intended for the end user and represent an issue with the software or the project's configuration. Error codes will be displayed in the client and/or server log and will show up in the format of `[ModuleName]-[ErrorCode]` where `[ModuleName]` would represent the NMS Module in context. WSW for Web Switching, JBOT for JBot configuration, and so forth. `[ErrorCode]` will uniquely define the error within each module.

If an error code is identified, further details for the error code can be found on the NMS Errors portal page. The portal page will be installed at the following location:

```
http://[WebLogic Host]:[port]/nms/errordocs/index.html
```

In some cases the error code will produce an Error Code error dialog with a short description for the error code. The dialog will also include buttons that allow you to review additional details about the code or to report the issue. The Report Issue... button will gather the environment's logs and initiate your default email client with the logs attached. The Details... button will bring up the extended details for the error code in your default web browser.

Error code short descriptions, which show up on the Error Code error dialog can be overridden by added an entry for the code in the `MessageCode_en_US.properties` file. Here is an example:

```
OmsErrorCodeException.TEST_EXAMPLE-1000 = This is a test message with
arguments [{0}] [{1}].
```

This text will be displayed in the Error Code error dialog, but will not replace the message displayed in the log file. The message override option should not be necessary. However, if a particular issue were to be introduced with a patch that results in the error code being displayed on a regular basis, the project may want to override the message to give its users further details on how to deal with the issue.

The override option also allows the error code descriptions to be translated to another language in case the need arises.

Performing a Java Stack Dump

If a Java application hangs, it is usually necessary to provide Oracle Support with a stack dump of the application to debug this issue. To add the stack dump to the log file, press **CTRL+Shift+D**.

> **Note:** If there are multiple applications open, a stack dump will be added for each.

Emailing the Log File

To email the log file, either select the environment's **Email Log Files...** from the environment **Help** menu or press **CTRL+Shift+M**, which will create a new email message (using your default email client) with the log file attached; the email can then be addressed and sent.

> **Notes:**
>
> - If there are multiple applications open, separate log files will be attached for each; any unneeded logs can be deleting from the email prior to sending it.
> - If for any reason the NMS environment is hung, the **Help** menu option will be unavailable, but the **CTRL+Shift+M** sequence will still continue to function. It is strongly advised that all the operators of NMS learn and utilize this feature if they ever encounter a hung environment. Oracle Customer Support will require these logs when a service request is created.

- If the menu is chosen, only the log file for the current application will be attached. If the **Ctrl**+**Shift**+**M** hotkey is used, then the log files for all current applications, plus NMSMonitor will be attached.

Isis Log Files

There are two types of Isis log files:

- The Isis startup log tracks configuration information along with other notable information that occurs before protos is completely started. The nms-isis start program starts isis (isis in turn starts protos) using the nohup command, which makes protos immune to hang-ups, such as exiting the terminal after starting Isis. The startup log is called `isis.yyyymmdd.HHmmss.log` and can be found in `$NMS_LOG_DIR/run.yyyymmdd.HHmmss.log`. If you cannot start Isis, check this log.
- The protos log contains log information for the running protos process. This file is site-specific, and the name is based on the site number of the machine on which protos is running. The log for the protos process can be found in `$NMS_LOG_DIR/run.isis/[site #].logdir/[site #]_protos.[date].[time].log`.
- When Isis is restarted, the old log files will be archived into the `$NMS_LOG_DIR/run.isis/[site#].logdir/old_log` directory. They will be automatically removed after the number of days specified by `$NMS_DAYS_TO_LOG` if/when Isis is restarted.

Oracle RDBMS Log Files

Many times, there is an error in an Application log file that points to some sort of database problem. DBService may log that at a certain time the database was unavailable to answer queries. Look in the database logs to find the answer. These logs can alert you to problems with the RDBMS configuration, software, and operations. Other instances of a dbservice (TCDBService, PFDBService, MBDBService) may also be configured and running. Each of these should be reviewed for errors.

Refer to the Oracle RDBMS documentation for locations and instructions for viewing Oracle RDBMS logs.

Operating System Log Files

Another place to look for problems is in the operating system logs. Refer to the operating system documentation for locations and instructions for viewing operating system logs (generally various forms of syslog, such as `/var/log/messages` for Linux).

It is generally recommended that syslog be turned on for a production system. In particular, the Oracle Utilities Network Management System uses the syslog to track fatal errors and log the start/stop time of every Oracle Utilities Network Management System-specific Isis process.

Entries like the following can be useful when trying to track down which application binary a particular Unix process ID belongs to:

- May 30 12:47:57 msp-pelin01 CES::corbagateway[26346]: my_address = (2/7:26346.0)
- May 30 12:48:00 msp-pelin01 CES::corbagateway[26346]: **INFO*** [corbagateway-26346] for [msp-pelin01] exiting....

Using the Action Command to Start a New Log File

There is also a feature that uses the Action command to start a new log file without stopping anything. This can be very useful in isolating a portion of the log file when recreating a problem. The command is:

```
Action any.<NMS_ISIS_process_name> relog
```

For example: `Action any.JMService relog`

The Action command can also be used to turn debug on and off for services or tools. This can also be used with the relog feature to better isolate debug for a particular user scenario.

The following command will turn debug on:

```
Action any.<service> debug 1
```

The following command will turn debug off:

```
Action any.<service> debug 0
```

Each Oracle Utilities Network Management System Isis (daemon/service/adapter) process typically supports facility specific debug that can be enabled to help track down issues with the facility in question. In general, you must consult with Oracle Support to get details on what facilities are currently available and what level to set them to for a given situation.

Examining Core Files

On Unix, if a process has either committed an error or over-taxed the system resources, the operating system will kill it rather than letting it take down the operating system. When this happens, the operating system dumps the contents of the memory occupied by the process into a file named "core." These files can sometimes be analyzed to better understand the reason for the failure.

Unix operating systems have a Unix user specific mechanism (`ulimit -c`) that determines how much disk space a core file can use. If this value is set to 0 or too low, the operating system may not be able to generate a useful core file. It is recommended that production NMS systems run with "`ulimit -c`" set to a sufficiently high value (typically "unlimited"). Note it is left up to NMS end users to monitor/manage any disk space used by NMS processes. These core files typically only have a very short shelf life of value (often a few days while any analysis might happen) after which they can be removed.

Normally, you should question the production of a core file to see if there are any extraneous reasons why the OS is dumping a process. If you do not find anything, retrieve the core file and analyze it.

> **Note:** see the Pre-Installation chapter of the *Oracle Utilities Network Management System Installation Guide* for information on core file naming conventions.

Core files are located in the `NMS_LOG_DIR/run.[service]` directory in the username that started services, or in the directory where a tool was started (usually the home directory of the user).

After performing a **`kill -USR2`** on a hung process, it can be useful to follow with `kill -abrt [pid]`

This will cause the process to dump core and the process will be dead.

> **Note**: Always use `kill -USR2` before `kill -abrt` because the `-abrt` option terminates the process. Make sure it is okay to terminate the process before attempting `kill -abrt`.

The command `file core` will generally (depending on the operating system involved) identify which process generated the core. Later core files can overwrite earlier core files. Renaming the core file to something like `core.[process]` can prevent this.

SMService can be set up to automatically find, rename, and consolidate core files into a single directory (`$NMS_LOG_DIR/SavedCores` by default). You can change what happens to core files captured by SMService by modifying the `sms-core-save` script.

When a tool or service cores, the investigation is helped by sending the stack trace in the incident report. A stack trace can be generated using the dbx (Solaris) or gdb (Linux) tool. The syntax is as follows:

Solaris:

```
dbx <path to binary directory> <path to corefile>
```

Linux:

```
gdb [path to binary directory] [path to corefile]
```

For example:

```
dbx $NMS_BASE/bin/JMService ~/run.JMService/core
```

Press the space bar until you get a prompt and then enter the following commands:

Solaris:

```
where
threads
```

```
dump
regs
quit
```

AIX:

```
where
thread
dump
registers
quit
```

Linux:

```
where
info threads
info locals
info all-reg
thread apply all where
```

Include the results of these commands in your incident report.

Searching for Core Files

To search for core files, complete these steps:

1. Search for core files with the find command:

   ```
   $ find . -name core* -exec ls -l {} \;
   ```

   Expected result:

   ```
   -rw------- 1 ces users 32216692 Oct 15 16:05 ./core
   ```

   This executes an "`ls -l`" on any files found in the tree starting from the current working directory. This should be done from the $NMS_HOME directory and (if it differs from $NMS_HOME) the $HOME directory.

   If a service cores, the core file can be found in the `$NMS_LOG_DIR/SavedCores` or (if SMService failed or is not configured with a CoreScript to detect and/or move the core file) the `$NMS_LOG_DIR/run.[service] directory`. Note that SMService will rename a service core file to `[hostname]-[service]-[date].[time].core` to minimize the chance of core files overwriting each other.

2. Type the following to determine where a core file came from:

   ```
   $ file ./core
   ```

   Below is a sample result from an AIX server:

   ```
   core: AIX core file fulldump 64-bit, JMService - received SIGBUS
   ```

   The core file referenced above is the result of a JMService core dump. The output gives:

   - the file name (which is always "core"),
   - which program/process the file came from (JMService), and
   - optionally, the message that the program received from the OS (SIGBUS).

3. Generally the most useful thing you can do is to identify what is called the core stack trace--the specific functions that were called (in order) leading up to the violation that caused the operating system to generate the core file. The stack trace is often a useful

piece of information that, if available, should be captured for later analysis. Details on navigating a core trace can be found later in this document.

4. Use the strings command to get some more information out of the file, if possible. Type:

```
$ strings core | head
```

Sometimes the messages returned, such as "Out of memory" or "I/O error," give an idea of what might have happened.

Identifying Memory Leaks with monitor-ps-sizes

The `monitor-ps-sizes` script monitors the size of processes to identify potential leaks. It performs periodic snapshots of all running processes and warns the user of any processes that have grown greater than the specified size. It supports the following command-line options:

Option	Description
-A	Log the command's output
-a <command>	Command to perform on process when generating a warning. You can pass the program's name and/or PID via #PID# and #PROGRAM#
-f <list of user names>	Monitor processes for this comma-separated list of users.
-G <number>	A warning about a process is guaranteed to be generated if the process exceeds this size. **Default:** 40000 (units reported by ps)
-g <number>	The growth factor that triggers a report. **Default:** 1.75 (floating point numbers greater than 1 are valid)
-l <line number>	The line number that specifies the stable size in the process-size log file. **Default:** 3 (line numbers begin counting with 1)
-n <program names>	A comma-separated list of program names to monitor
-O <number>	The maximum number of seconds to retain log files. **Default:** 172800 (seconds) if 0, old log files are not erased.
-P <number>	The minimum number of seconds to wait between warnings. **Default:** 0 (seconds)
-p <number>	The number of seconds to wait between snapshots. **Default:** 3600 (seconds)
-R <number>	The minimum process size that can be reported. **Default:** 5000 (units reported by ps)
-s <email subject line>	The subject line to use to title email warnings about processes that are too big. **Default:** "process size warning for prod_model"
-u <email names>	A comma-separated list of users to email when there are processes warnings. **Default:** no email sent.

For example, to monitor JMService and MTService for user "nms" when either gets larger than 500 MB or grows by 10 percent, use:

```
monitor-ps-sizes -n MTService,JMService -f nms -R 500000 -g 1.1
```

Validating the WebLogic Caches with NMS Services

The NMS application deployed to WebLogic (nmsejb.ear), contains various caches that are used to lessen the load on the NMS services. Normally they are kept in sync automatically. However, there are certain circumstances when it is desirable to force the system to refresh the system.

The general command is:

```
Action any.publisher* ejb refresh
```

This command causes the system to reload the configuration and forces the client to re-request all data that it is currently displaying. This puts significant load on the system, so it should only be done when necessary in a production environment.

The following commands do not put much load on the server, so they are safer to call in a production environment:

- If only the viewer symbology needs to be reloaded, use this command:

  ```
  Action any.publisher* ejb reload_symbology
  ```

- To validate the event cache use:

  ```
  Action any.publisher* ejb resync
  ```

If there were any changes due to the re-synchronization, they will be logged to the WebLogic log file.

Monitoring EclipseLink Related Database Transactions

EclipseLink provides an Object-Relational Mapping (ORM) structure that NMS uses in some cases to manage database data. EclipseLink is not used by all of the NMS tools, but is heavily used by Web Switching and Web Safety. In some cases, project configuration can cause database related issues that are not obvious to the implementer via the standard log messages. When additional details about the database queries are required, the EclipseLink debugging level can be turned on and even redirected to a specific log. The log file will be output on the same server where your WebLogic installation is running. Since the debug can place additional strain on an environment, it is strongly advised that any EclipseLink related debug not be generated in a production environment unless absolutely necessary to diagnose an issue. Projects should always run their production environments with the standard EclipseLink configuration file included with NMS.

The following directions should be used to define your project version of the EclipseLink configuration file.

1. In your [project]/jconfig directory, create a subdirectory structure as follows:

   ```
   [project]/jconfig/override/fwserver.jar/META-INF/
   ```

2. Copy the Product version of the persistence.xml file. This file can be found in the product/jconfig/override/fwserver.jar/META-INF/ directory.
3. Save the file to the META-INF directory that you created in step 1.
4. You should find the entries like the following commented out in the configuration file.

   ```
   <property name="eclipselink.logging.logger" value="ServerLogger"/>
   <property name="eclipselink.logging.file" value="/users/nms1/
   nmslogs/eclipselink.out" />
   ```

   Uncomment the entries and set the `logging.file entry` to a directory that exists on the server where your WebLogic instance is running. If this directory does not exist, the NMS application deployed to WebLogic will not start.

5. Find the following line:

   ```
   <property name="eclipselink.logging.level" value="SEVERE"/>
   ```

   Set the value to FINE.

   ```
   <property name="eclipselink.logging.level" value="FINE"/>
   ```

   Valid values include ALL, FINEST, FINE, CONFIG, INFO, WARNING, SEVERE and OFF. Use caution using **FINEST** or **ALL** because they generate a large amount of debug information. For diagnosing most issues, FINE is a good place to start.

6. Locate the following line:

   ```
   <jta-data-source>jdbc/intersys</jta-data-source>
   ```

   Change the jta-data-source to match the config.datasource value in $NMS_CONFIG/jconfig/build.properties. For example, if config.datasource is set to jdbc/intersys/build23, then change the line to read:

   ```
   <jta-data-source>jdbc/intersys/build23</jta-data-source>
   ```

7. When the changes are complete, build a new cesejb.ear file and deploy this to your WebLogic instance.

Performance Testing

It is helpful to have certain debug options enabled when doing performance/scalability testing in order to analyse issues after the test. The following debug is recommended.

In system.dat:

```
corbagateway -pgtiming on -debug GATEWAY_MESSAGE 1
JMService -debug API 1 -debug TIMING 2
DDService -debug MESSAGES 1
```

And to turn on Agent debug in WebLogic:

```
Action any.publisher* ejb debug com.splwg.oms.ejb.session.Agent=DEBUG
```

It may also be desirable to enable EJB Logging (described above) or to enable ProxyInvocationHandler debug in the clients. ProxyInvocationHandler debug outputs 1 line for each remote method call made by the client to the client log (for example, `WebWorkspace.log`). To enable ProxyInvocationHandler debug for individual users at run time, run the following command (changing [userid] to the user's login id):

```
Action any.publisher* ejb client <userid> debug
```

To enable ProxyInvocationHandler debug for all users, add the following line to `$NMS_CONFIG/jconfig/global/propeties/log4j.properties`:

```
log4j.category.com.splwg.oms.client.util.proxy.ProxyInvocationHandler=
DEBUG
```

It is also recommended to run `st-call-rate` for the duration of the test. This script periodically queries the NMS database to report information about calls/incidents, outages, and device operations. The recommended options for this script are:

```
st-call-rate -i 60 -m <minutes>
```

With <minutes> specifying the number of minutes to run. For example, to run for 4 hours and output to a file named `call_rate.out`, run the following:

```
nohup st-call-rate -i 60 -m 240 > call_rate.out 2>&1 &
```

Run `st-call-rate -h` for the most current documentation of command-line options and columns of output from this script.

The following scripts can be run on log files to summarize timing results. Each script takes a log file to read as a parameter.

- **`st-pg-report`**: Summarizes the process group timing in service logs. Run this on the corbagateway log or any other service log that pgtiming was enabled on.
- **`st-agent-report`**: Summarizes the Agent timing debug in the WebLogic log. Run this on the file configured to get log4j messages for com.splwg.oms.* (typically the `.out` file).
- **`st-client-report`**: Summarizes the ProxyInvocationHandler timing debug in the client log.

The output of the above scripts is very similar with one line of output per API or method call and the following columns:

- low - the lowest duration for the api/method call
- median - the median duration for the api/method call
- avg - the average (mean) duration for the api/method call
- high - the highest duration for the api/method call
- total - the total duration of all calls to this api/method
- count - the number of invocations of this api/method
- api/method - the name or identifier of the api/method

After a performance test, grep the service logs for any of the following strings:

"^Dump requested"

"Congested"

"time warp"

"ORA-"

"ERROR"

Note that "ERROR" can still appear quite a bit during performance testing for situations like a user (automated or real) performing an action on an event that either grouped into another event or has been canceled or completed.

In the WebLogic log4j log (typically the .out file) grep for:

- "ORA-"
- "ERROR"

Other Troubleshooting Utilities

Using the JMS API Command Line Utility to Manually Change a Job

The JMS API command line utility (`jms-api`) provides a restricted set of options to modify an event when the event cannot be changed with the NMS user interface. It is primarily intended for cleaning up stranded events or other issues where normal NMS functionality will not work. Indiscriminate use of `jms-api` for frequent/high volume activity in conjunction with NMS operation can have a negative performance impact on NMS and is not recommended

Standard Usage:

```
$ jms-api [option] [event]
```

where

- `[option]` is the jms-api option
- `[event]` is an event handle of the form 800.[event#] (for example, 800.10257)

- To return the `jms-api` usage options and arguments:

```
$ jms-api
```

- To complete an event:

  ```
  $ jms-api complete [event] "[comments]"
  ```

  Note: does not allow an RDO still affecting customers to be completed.

- To cancel an event:

  ```
  $ jms-api cancel [event] "[comments]"
  ```

  Note: does not allow an RDO still affecting customers to be canceled.

- To complete a Master Switching Job or Planned Outage:

  ```
  $ jms-api swplan_complete [event] "[comments]"
  ```

  Note: does not allow an active Planned Outage or a Master Switching Job with active Planned Outage(s) to be completed

- To cancel ("reschedule") a Master Switching Job:

  ```
  $ jms-api swplan_cancel [event] "[comments]"
  ```

- To remove association between event and switch sheet:

  ```
  $ jms-api remove_assoc [event] "[comments]"
  ```

- To change the estimated restore time of a job:

  ```
  $ jms-api set_est_rest_time [event] "[comments]"
  ```

  Note: *<time>* must be a valid ISO-8601 date/time string (for example, *2020-02-27T15:30*).

- To set the external id of a job:

  ```
  $ jms-api set_external_id [event] [value]
  ```

- To set the customers out for a job:

  ```
  $ jms-api set_cust_out [event] [value]
  ```

- To set the trouble code of a job:

  ```
  $ jms-api set_trouble_code [event] [value]
  ```

  Note: does not modify the calls on a job; `[value]` must be a valid numeric trouble code.

- Alternative API for completing an event

  ```
  $ jms-api complete2 [event]
  ```

  Note: The `complete2` parameter does not work for a Master Switching Job or Planned Outage. It does not validate that an RDO event is restored before completing it.

Oracle Support Information

Support Knowledgebase

Additional troubleshooting information can be found on My Oracle Support at:

```
http://support.oracle.com
```

Contacting Oracle Support

For support please contact Oracle Support at:

```
http://www.oracle.com/support/index.html
```

Chapter 13

NMS Data Quality

The NMS installation provides a data qualification script that can be run before attempting to load historical data to an external system. This script checks for some of the most common data issues and mismatches in the NMS system and lists the errors in your data.

To output the data validation to a file (dq.out), run:

```
ISQL < ${NMS_SQL_FILES}/OUA_Data_Qualification.sql 2>&1 | tee dq.out
```

Follow the output to correct any data issues, or they will potentially generate millions of errors and slow down your initial load.

Some of the most common errors are:

- **Mismatches between Customers and events or calls.**

  This can happen if you ran your NMS before history was added to the customer records. The CES_CUSTOMERS_HISTORY defaults to a birth date of January 1, 2000, but is sometimes changed by the project. The CU_CUSTOMERS, CU_SERVICE_POINTS, CU_SERVICE_LOCATIONS, and CU_METERS may all have different birth times, since those are populated from an external system.

  Event and call history, therefore, may reference customers that are not listed as active in one or more of these tables. Either import the correct historical customer data from an external system, or back-date the first CES_CUSTOMERS_HISTORY and CU_% table records' birth dates to the time of the earliest event or call.

- **Mismatches between control zones and events or calls.**

  This can happen if you ran your NMS before history was added to the control zones tables. In the unlikely event you have control zone history in an external system, update inactive records in CONTROL_ZONES and CONTROL_ZONE_STRUCTURES to match, or back-date the birth dates for the first records.

- Mismatches between the CES_CUSTOMERS_HISTORY table and CU_CUSTOMERS and related tables.

  This can happen if the CU_CUSTOMERS, CU_METERS, CU_SERVICE_POINTS, and the CU_SERVICE_LOCATIONS tables were created before or after the CES_CUSTOMERS_HISTORY records, or if the CES_CUSTOMERS table is created using unorthodox customer mapping.

It is expected that:

- CES_CUSTOMERS.cust_id matches CU_CUSTOMERS.cust_id
- CES_CUSTOMERS.meter_id matches CU_METERS.meter_id
- CES_CUSTOMERS.serv_loc_id matches CU_SERVICE_LOCATIONS.serv_loc_id

  These three columns match the corresponding CU_SERVICE_POINTS fields.
- CES_CUSTOMERS.account_number matches CU_SERVICE_LOCATIONS.serv_account_number

Chapter 14

Setting Up NMS for Oracle Utilities Analytics

To use a single Oracle Utilities Analytics instance with multiple NMS environments, you need to indicate the data source for each NMS environment.

```
INSERT INTO CES_PARAMETERS (APP, ATTRIB, VALUE) VALUES
  ('BI', 'DATA_SOURCE_INDICATOR', '000004');
```

Chapter 15

User Authentication

This chapter describes how to configure authentication of users for the Oracle Utilities Network Management System (NMS) applications.

- Overview of Authentication
- Configuring the WebLogic Security Realm
- Configuring Authentication Using WebLogic Internal Users/Groups
- Configuring Authentication Using an Active Directory Provider
- Configuring Authentication Using an OpenLDAP Provider

Overview of Authentication

To use NMS, a user has to be configured for both authentication and authorization.

Authentication (user names and passwords) for Oracle Utilities Network Management System is handled by WebLogic, and is accomplished by configuring authentication providers in WebLogic's default security realm. This is a simplification from previous releases, where user names and passwords were kept in database tables, or where LDAP or Active Directory information had to be configured in SQL files.

Authorization (what applications a user is allowed to use, with what role or user type, or whether the user is allowed to login to the NMS at all) is handled by the Configuration Assistant. See chapter 16 of the Oracle Utilities Network Management System User's Guide for more information on the use of the Configuration Assistant.

Most installations will want to configure WebLogic to use an external authentication source, such as Active Directory or LDAP. These servers are often readily available on most corporate networks, they provide advantages for enforcing security policies (*e.g.*, password complexity and aging), and the login names and passwords are already familiar to the end users. In the case that a more simple solution is required, WebLogic internal users and groups can be used to authenticate against the NMS, although this is not recommended for production environments.

Any user that appears in the users and groups in WebLogic's default security realm tab can be configured to login to NMS, with the following conditions:

- The user must exist in a group that has access to the nms roles as configured under Configuring NMS Security Roles.
- The user must be added to **NMS** through the Configuration Assistant. This will add the user to the CES_USER and USER_PERMISSIONS tables. This will also add the user to the ENV_ACCESS table for each configured product (for example, Web Workspace, Web Call Entry, etc.).

Without both of these conditions being met, the application will return that the user is unauthorized.

Configuring the WebLogic Security Realm

1. Login to the WebLogic Administration Console
2. In the Domain Structure pane, click on Security Realms.
3. Click on the default security realm (typically called myrealm).
4. Click on the Providers tab.
5. Click on DefaultAuthenticator.
6. Change Control Flag so it is set to OPTIONAL.

Configuring Authentication Using WebLogic Internal Users/ Groups

The following steps can be used to create users and groups directly in the WebLogic default security realm.

1. Login to the WebLogic Administration Console
2. In the Domain Structure pane, click on Security Realms
3. Click on the default security realm (typically called myrealm).
4. Click on the Users and Groups tab, and then click on the Groups tab.
5. Click on the New button to create a new group.
6. Enter the following group properties:

   Name: *nms-user* (or other group with nms configured roles). nms-service is also required for the "publisher.ejb-user" configured in the build.properties file.

   Description: asGroup membership for NMS login.

   Provider: DefaultAuthenticator
7. For each user to be created, click on the **Users** tab, and press the **New** button to create a new user. Enter the following user properties:

   Name: juser

   Description: Joe User

   Provider: DefaultAuthenticator

   Password: *****

   Confirm Password: *****

   > **Note:** User names must be unique. Passwords must contain at least one special character.
8. For each user created, click on that user name in the list of users. Click the **Groups** tab, select the **nmsuser** group from the list of available groups, and move it to the **Chosen** list by using the **>** button. Click **Save**.

It is recommend that the credentials used to authenticate SwService (and other external adapters) be configured in the internal DefaultAuthenticator. Additionally, this provider should be configured to be the first location used to avoid potential performance problems with latency issues on network based directory services.

Configuring Authentication Using an Active Directory Provider

This section provides an example for how to connect WebLogic to an Active Directory. The specifics of your Active Directory domain may differ from the example given, so consult with your Active Directory administrator to find the correct values, and refer to the WebLogic documentation for specifics on each option.

1. Login to the WebLogic Administration Console.
2. In the **Domain Structure** pane, click on **Security Realms**.
3. Click on the default security realm (typically called myrealm).
4. Click on the **Providers** tab and click the **New** button.
5. Provide a name for the provider (for example, "nms-provider"), and select **ActiveDirectoryAuthenticator** as the type.
6. Click the name of the newly created provider.
7. Under the **Configuration** tab, select the **Common** tab, and set **Control Flag** to **Optional**.
8. Click **Save**.
9. Under the **Configuration** tab, select the **Provider Specific** tab, and set desired values that match your Active Directory configuration.

   Examples:

   Connection

   Host: server.example.com

   Port: 389

   Principal: cn=Administrator,cn=Users,dc=example,dc=com

   Credential: (the password used to connect to the account defined by Principal)

   Users

   User Base DN: cn=Users,dc=example,dc=com

   User From Name Filter: (&(samAccountName=%u)(objectclass=user))

   User Name Attribute: samAccountName

   User Object Class: user

   Groups

   Group Base DN: cn=Groups,dc=example,dc=com

   Group From Name Filter: (&(cn=%g)(objectclass=group))

10. Click **Save**.
11. In the **Change Center**, click **Activate Changes**.
12. Restart the AdminServer.
13. **IMPORTANT:** Verify that the users and groups from the Active Directory are configured by looking at the Users and Groups tab under the default security realm. If not, adjust the configuration.

Configuring Authentication Using an OpenLDAP Provider

This section provides an example of how to connect WebLogic to an OpenLDAP server. The specifics of your OpenLDAP directory may differ from the example given, so consult with your LDAP administrator to find the correct values, and refer to the WebLogic documentation for specifics on each option.

1. Login to the WebLogic Administration Console.
2. In the **Domain Structure** pane, click on **Security Realms**.
3. Click the default security realm (typically called myrealm).
4. Click the **Providers** tab and press the **New** button.
5. Provide a name for the provider (for example, "nms-provider"), and select **OpenLDAPAuthenticator** as the type.
6. Click the name of the newly created provider.
7. Under the **Configuration** tab, select the **Common** tab, and set **Control Flag** to **Optional**.
8. Click **Save**.
9. Under the **Configuration** tab, select the **Provider Specific** tab, and set desired values that match your LDAP Directory configuration.

Examples:

Connection

Host: server.example.com

Port: 389

Principal: cn=Manager,dc=example,dc=com

Credential: (the password used to connect to the account defined by Principal)

Users

User Base DN: ou=Users,dc=example,dc=com

User from Name Filter: (&(uid=%u)(objectclass=inetOrgPerson))

User Name Attribute: uid

User Object Class: inetOrgPerson

Groups

Group Base DN: ou=groups,dc=example,dc=com

Group From Name Filter: (&(cn=%g)(objectclass=groupOfNames))

10. Click **Save**.
11. In the **Change Center**, click **Activate Changes**.
12. Restart the AdminServer.
13. **IMPORTANT:** Verify that the users and groups from the LDAP server are configured by looking at the Users and Groups tab under the default security realm. If not, adjust the configuration.

Two-Factor Authentication

NMS applications can, optionally, use two-factor authentication to further enhance security Email, SMS messages, or third party providers can be configured. Two-factor authentication is configured by modifying Multifactor.xml and the dialog DLG_MULTIFACTOR.xml.

The product version of Multifactor.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--  Configuration file for sending Multifactor authentication
emails. -->
<multifactor>
    <config digits-to-send="6" max-attempts="3"
expiration-seconds="600" max-requests="3"/>
    <provider type="Email"
class="com.splwg.oms.common.ejbaccess.EmailMultifactorAuthenticator">
       <!-- The following is a select statement that returns the email
address send the code to
            The default value assumes that the full_name contains the
email address of the user -->
        <!-- another possible value is: "select ? || '@example.com'
from dual" -->
        <to sql="select full_name from ces_user where user_name= ? and
active = 'Y'"/>
        <from>nobody@oracle.com</from>
        <subject>Requested Oracle NMS Validation Code</subject>
        <body type="text/plain">Here is the requested Oracle NMS
Validate Code: ${CODE}</body>
    </provider>
    <provider type="SMS"
class="com.splwg.oms.common.ejbaccess.EmailMultifactorAuthenticator">
       <!-- The following is a select statement that returns the email
address of the gateway to send the code to -->
        <to sql="select ? || '@smsgateway.example.com' from dual"/>
        <from>nobody@oracle.com</from>
        <body type="text/plain">The requested Oracle NMS Code:
${CODE}</body>
    </provider>
</multifactor>
```

Two-factor authentication is enabled by assigning users to the **NmsMultifactor** role. Users in the **nms-twofactor** group will have two-factor authentication enabled by default.

Third party providers can be configured by extending com.splwg.oms.common.ejbaccess;.MultifactorAuthenticator. Then, add a configuration section for the new class in Multifactor.xml. The provider type should match the value of the combo box in the dialog.

Chapter 16

Fault Location, Isolation, and Service Restoration Administration

This chapter describes how to configure and administer Fault Location, Isolation, and Service Restoration (FLISR). It includes the following topics:

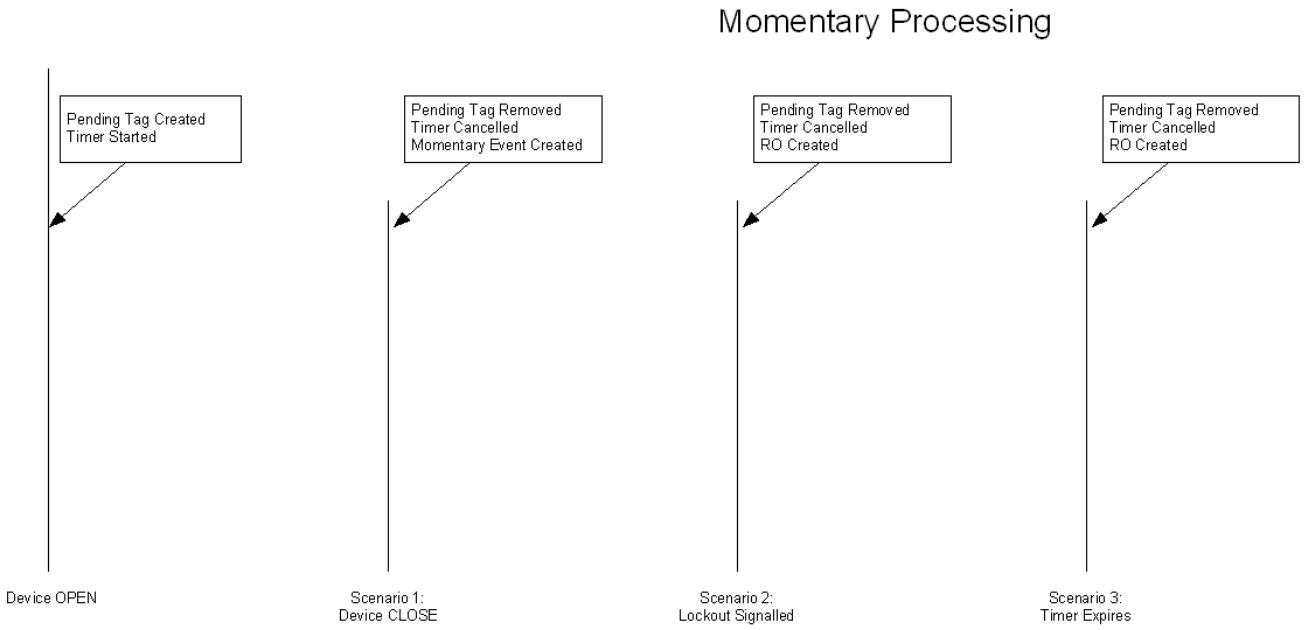
- Introduction
- Fault Location, Isolation, and Service Restoration Timeline
- Software Architecture Overview
- Configuring Classes and Inheritance
- SRS Rules
- High Level Messages
- Troubleshooting

Introduction

The intended audience for this document is the system administrators responsible for maintaining the Oracle Utilities Network Management System.

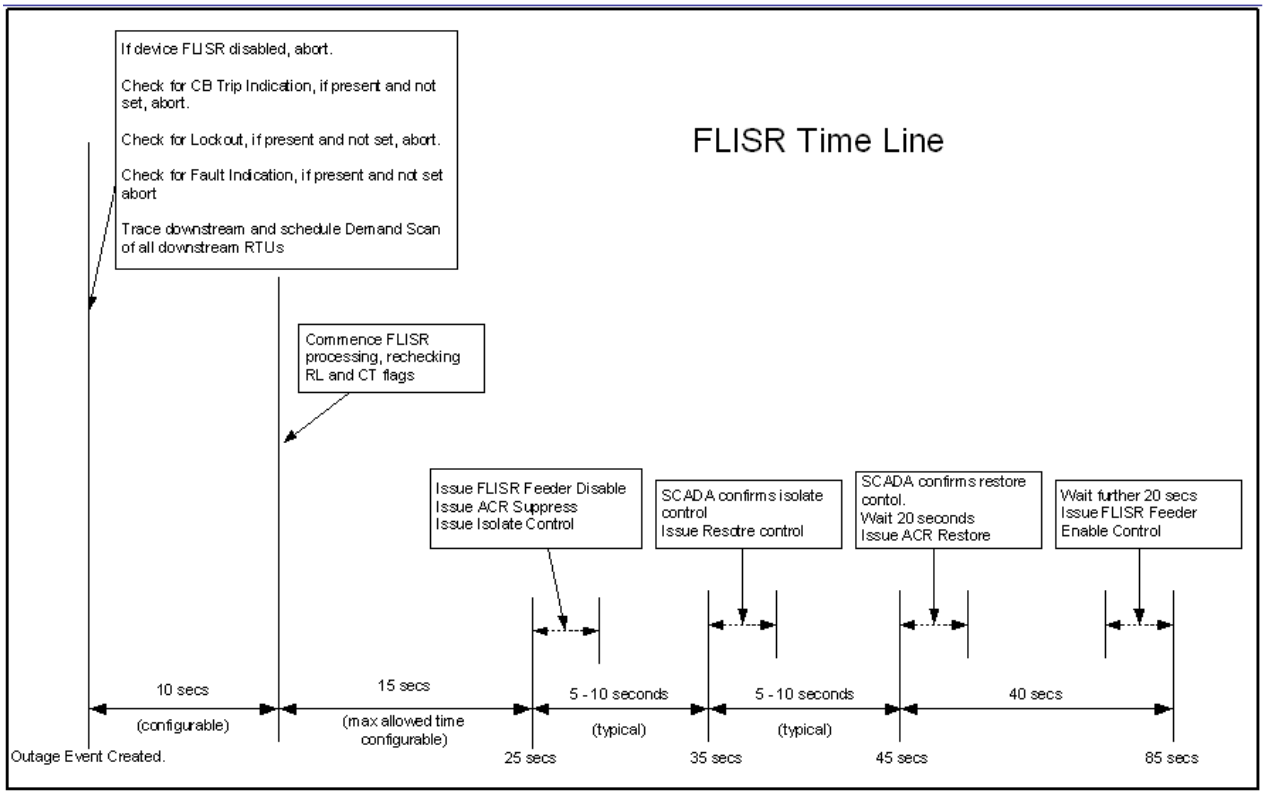
Fault Location, Isolation, and Service Restoration Timeline

These figures show the sequence of events in a Fault Location, Isolation, and Service Restoration scenario. The following figure shows the various scenarios in the momentary processing.



Note: RO is created only if customer supply nodes are de-energized as a result of the operation.

Once an RO is created, the Fault Location, Isolation, and Service Restoration processing sequence shown in the following figure is initiated.



The control sequence (starting at around 25 seconds) is only performed in automatic mode. In manual mode an operator must initiate the control sequence.

Timings in the above diagram are only indicative. Actual values will depend on the complexity of the solution required and the responsiveness of the isolate/restore controls sent to SCADA. The following timings are deterministic:

- The delay allowed for demand scans. This is configurable and defaults to 10 seconds
- The maximum time allowed for the solution in automatic mode. This is configurable and defaults to 15 seconds. If the solution takes longer to solve than this time, Fault Location, Isolation, and Service Restoration will not automatically execute the control sequence. The option for an operator to manually initiate the control sequence is preserved though.
- Maximum time allowed for automatic operations after the lockout is: Demand scan delay + 15 seconds (25 seconds in the default configuration).
- Wait times for Auto-Reclose operations. These are 20 seconds.

Software Architecture Overview

This section describes the role of various software components in implementing the Fault Location, Isolation, and Service Restoration functionality:

Component	Description
DDService	Tracks SCADA measurements, device operations and Conditions. DDService is the starting point for Fault Location, Isolation, and Service Restoration events. When a device trips, a pending operation is created. When the lockout occurs a completed device operation is sent to MTService. If the breaker is able to reclose – only a momentary event is created. DDService is also responsible for executing Fault Location, Isolation, and Service Restoration switch plans, both in manual and automatic mode. In manual mode the request to execute the switch plan can be initiated by the operator from the Switch Sheet Editor tool. In automatic mode the Fault Location, Isolation, and Service Restoration sub-system requests the switch sheet execution by DDService
PFService	The core of Fault Location, Isolation, and Service Restoration functionality. It contains most of the Fault Location, Isolation, and Service Restoration sub-system. Its initial task is to process device operations from DDService and determine the extent of energization changes in the model. These changes are also calculated by MTService and propagated to JMService for outage processing. If the device operation is a trip, the Fault Location, Isolation, and Service Restoration sub-system will perform an initial trace to initiate a demand scan of affected RTUs. The bulk of Fault Location, Isolation, and Service Restoration processing is triggered by JMService deciding that event has de-energised customers. In this scenario JMService instructs PFService to initiate Fault Location, Isolation, and Service Restoration processing. PFService then calculates the various isolate and restore scenarios and populates the database tables with the solutions.
JMService	Receives notifications from MTService about changes in energization on the network. JMService will determine if these changes de-energises customers and if so creates an outage event and informs PFService that Fault Location, Isolation, and Service Restoration processing of that event is required.

Component	Description
WorkAgenda	Monitors notifications from JMService about the creation, update and completion of events. WorkAgenda is configured to highlight Fault Location, Isolation, and Service Restoration events in various ways: • Events detected as potential Fault Location, Isolation, and Service Restoration events are highlighted with a yellow background. The background stays yellow until a Fault Location, Isolation, and Service Restoration solution is found or a further determination indicates that the event cannot be considered an FLISR event (e.g., all restoring switches or feeders are Fault Location, Isolation, and Service Restoration disabled) • Events for which a viable Fault Location, Isolation, and Service Restoration solution is found are highlighted with a pink background. • Events for which a Fault Location, Isolation, and Service Restoration solution is found, but the solution includes overloads on restoring feeders, are highlighted with a light blue background.
FLISR	Provides a summary of the Fault Location, Isolation, and Service Restoration solution for an event. If an event is found to have a Fault Location, Isolation, and Service Restoration solution, the operator can examine the details of that solution by using this tool. This tool primarily reads the database tables to determine the solution information calculated by PFService. The operator can also manually write, append and/or overwrite the generated switch plan.
Switching	Once a solution is found for the Fault Location, Isolation, and Service Restoration event, a switch plan can be created to execute the solution. The switch plan can be created (and executed) automatically, or it can be created manually. In either scenario the switch plan can be viewed from the Switch Sheet Editor. In manual mode the operator can request that DDService execute the plan. In both manual and automatic mode the operator can watch the results of DDService performing a switch plan execution.

Configuring Classes and Inheritance

Fault Location, Isolation, and Service Restoration utilizes standard class names to determine various features in the model. Devices in a model can be configured to the Fault Location, Isolation, and Service Restoration classes using class inheritance.

The following table lists the classes supported by Fault Location, Isolation, and Service Restoration:

Class Name	Purpose
flisr_cb	Set of SCADA devices that are protective. These are the SCADA devices, which are generally at the feeder head, that can trip when a fault is detected.
flisr_sectionalizer	Set of protective or non-protective devices that are SCADA controllable. These devices: • Might have fault indicators on them in order to give better indication of fault locations on the feeder. • Will be considered for isolate and restore devices. • These are not feeder head devices.
flisr_fuse	Set of non-SCADA protective devices. These are considered when determining loads and limiting devices.
flisr_load	Set of devices that are loads on the network – typically distribution transformers.
flisr_cogen	Set of devices on the network that provide additional supply.
conductor	Set of conductor classes on the network. These are considered when determining limiting devices.
block_flisr	Condition classes. These define tags and conditions that automatically prohibit Fault Location, Isolation, and Service Restoration operations on a device.
pf_sef_unrel_switch	Set of devices on the network which cannot be trusted for their sensitive earth fault reporting.
pf_sef_semirel_switch	Set of device which can reliably report the absence of the sensitive earth fault but cannot reliably report the presence of the sensitive earth fault.

Single Phase FLISR Operation

For single phase FLISR operation, phase-operable (un-ganged) SCADA controllable devices are needed. Phase-operable SCADA feeder head circuit breakers should be modeled as scada_rack_ungang_circuit_breaker class and should be inherited from flisr_cb class. The mid-feeder phase-operable SCADA controllable switches should be modeled as scada_recloser_ungang class and should be inherited from flisr_sectionalizer class.

Restoring Substation Bus on Voltage Loss

To enable the FLISR to process the sub-station bus faults, sub-station switches should be inherited by flisr_sectionalizer. These devices should also be configured as protective devices for FLISR to use them for isolation and restore.

Modeling Information

In order to determine loads and limiting devices Fault Location, Isolation and Service Restoration needs to know loading information about devices. This data is obtained from the Power Flow model tables.

If Fault Location, Isolation and Service Restoration is being run in kVA Mode, and no other DMS applications are being utilized, a reduced Power Flow model can be populated. The minimum data requirements of this model are:

PF_SOURCES Table

h_cls	INTEGER	Class number of device
h_idx	INTEGER	Index number of device
base_voltage	FLOAT	Nominal voltage of device
<model attributes>		Common columns for model managed tables

PF_XFMRS Table

h_cls	INTEGER	Class number of device
h_idx	INTEGER	Index number of device
pri_base_voltage	FLOAT	Nominal voltage of transformer primary side
sec_base_voltage	FLOAT	Nominal voltage of transformer secondary side
rating_a	FLOAT	Rating of phase A of transformer
rating_b	FLOAT	Rating of phase B of transformer
rating_c	FLOAT	Rating of phase C of transformer
<model attributes>		Common columns for model managed tables

PF_XFMRS_TANKS Table

h_cls	INTEGER	Class number of device
h_idx	INTEGER	Index number of device
<model attributes>		Common columns for model managed tables

PF_LOADS Table

h_cls	INTEGER	Class number of device
h_idx	INTEGER	Index number of device
kva_rating_a	FLOAT	Rating of phase A of transformer
kva_rating_b	FLOAT	Rating of phase B of transformer
kva_rating_c	FLOAT	Rating of phase C of transformer
utilfac	FLOAT	Utilization factor of loads rated value
conforming	INTEGER	1 = conforming (scalable) load 0 = non-conforming
profile_id	STRING	Link to PF_LOAD_INTERVAL_DATA that describes the loads usage profile
<model attributes>		Common columns for model managed tables

PF_SWITCH Table

h_cls	INTEGER	Class number of device
h_idx	INTEGER	Index number of device
amp_limit	FLOAT	Rating of ampere rating of device
<model attributes>		Common columns for model managed tables

NETWORK_COMPONENTS Table

h_cls	INTEGER	Class number of device
h_idx	INTEGER	Index number of device
wire_size	STRING	Key to PF_LINE_CATALOG & PF_LINE_LIMITS table (required by conductors only)
<model attributes>		Common columns for model managed tables

PF_LINE_LIMITS Table

catalog_id	STRING	Key for table
season	INTEGER	season for rating
type	INTEGER	1 = Normal Rating 2 = Emergency Rating
phase	INTEGER	Phase rating applies to 0 = Phase A 1 = Phase B 2 = Phase C
limit	FLOAT	Ampere limit for conductor
Tablecatalog_id	STRING	Key for table
<model attributes>		<model attributes> are the list of common core table columns that are required for all model managed tables, they include: • h_cls • h_idx • partition • birth • birth_patch • death • death_patch • ptn_date • active

Even though Fault Location, Isolation and Service Reforestation uses a reduce model, the model loader in the NMS services is expecting to see a full schema. Therefore for a kVA only implementation, the full Power Flow model schema must be created and unused fields should be populated with reasonable defaults. A reasonable default is required to allow the model loader to pass the data verification checks.

SRS Rules

FLISR SRS rules can be found in the **Configuration Assistant's Event Management Rules** tab. See the Configuration Assistant chapter of the *Oracle Utilities Network Management System Users Guide* for information on accessing SRS rule information.

High Level Messages

PFService accepts the following High Level messages:

```
Action any.PFService <command> <arguments>
```

Where:

Command	Arguments	Description
debug FLISR	<N>	Sets the debug level: 0 = off 1 = demand scan & timing info 2 = Trace 3 = Detailed Information regarding solution 4 = Full debug
flisr kva_tolerance	<N>	Sets the capacity tolerance to allow. Where <N> is the new tolerance in kVA
flisr base_flows		Outputs the base conductor flow information
flisr ties		Outputs the ties (open) point summary
flisr alarms		Forces a check for the Fault Location, Isolation, and Service Restoration disabled device alarms
flisr check	ON/OFF	Toggle Fault Location, Isolation, and Service Restoration check mode on/off
flisr reload		Reload measurement configuration
flisr dump		Write internal data structures into log

Troubleshooting

The following high-level messages can be used to turn timing and demand scan information on/off. This is useful in determining that Fault Location Isolation Service Restoration is scanning the correct RTUs and that timing goals are being achieved.

To turn on the messages:

```
Action any.PFService debug FLISR 1
```

To turn off the messages:

```
Action any.PFService debug FLISR 0
```

Chapter 17

Distribution Management Application Configuration

This chapter provides an overview of the configuration and maintenance of Oracle Utilities Distribution Management System applications. It includes the following topics:

- Configuring Power Flow
- Power Flow Rules

For Distribution Management System installation instructions, see the *Oracle Utilities Network Management System Installation Guide*.

Configuring Power Flow

PFService (Power Flow Service)

The main application that runs the majority of the Oracle Utilities Network Management System Distribution Management business logic is the Power Flow service. If your environment will be running any applications listed in the previous section (except Web Switching and FLISR), you must add the Power Flow Service as a system service by updating the `$NMS_HOME/etc/system.dat` file. There are 3 main sections where this service needs to be defined: the service, program and instance sections. See the `$NMS_BASE/templates/system.dat.template` file for examples of how to configure the Powerflow Service. Search for PFService in the file and copy those lines to `$NMS_HOME/etc/system.dat` file. Make sure all lines are uncommented so that they are active. You must restart the system services in order for the Powerflow Service to be properly monitored by SMService.

The command line options for PFService are:

- **pfdbs**: Use a dedicated database connection, rather than the common pool. Requires a corresponding PFDBService instance to be defined in system.dat

Non-Converged Islands

When PFService encounters apparent model errors that preclude a solution for an island, the island is marked as "Non-Converged" and the Power Flow solution attempt is stopped. The island at this point is ‘disabled.’

To output the list of disabled islands to the PFService log file:

```
Action any.PFService dump_disabled_islands
```

When the model has been rebuilt with data to solve the error, you may re-enable the island:

```
Action any.PFService reenable_island <source alias or handle>
```

Power Flow Inheritance

PFService uses class inheritance to determine which devices in the model have certain properties, or can be considered for certain actions. The classes used for inheritance are:

- **pf_backfeed_detect:** the set of classes that should be checked for backfeed power violations.
- **pf_dist_gen:** the set of classes that should be treated as distributed generation units.
- **pf_capacitors:** the set of classes that should be treated as shunt capacitors.
- **pf_feeder_cls:** the set of classes that should be treated as feeder head circuit breakers.
- **pf_lines:** the set of classes that should be considered as conductors with power flow attribution (e.g., catalog data).
- **pf_loads:** the set of devices that should be considered as load points for powerflow analysis (e.g., load interval and ratings data).
- **pf_transformers:** the set of devices that should be considered as transformers with power flow attribution.
- **pf_regulators:** the set of devices that should be considered as voltage regulators with power flow attribution. Regulators will be similar to transformers except rated in amps and violations will be flagged in terms of amps.
- **pf_sub_xfmr:** the set of devices that should be considered as substation transformers with power flow attribution.
- **pf_isolate_switch:** the set of classes to consider for isolate operations in suggested switching.
- **pf_restore_switch:** the set of classes that indicate which devices should be considered as tie points within the model.
- **pf_scada_isolate_switch:** the set of classes to consider for SCADA isolate operations in suggested switching.
- **pf_scada_restore_switch:** the set of classes that indicate which devices should be considered as SCADA enabled tie points within the model.
- **pf_opf_enabled_devices**: the set of classes which needs to be selected by default to take part in the volt-var operations in the study mode.
- **pf_opf_excluded_devices**: the set of classes which should not appear in the controllable devices list of study mode volt-var gui.

Power Flow Rules

Oracle Utilities Network Management System Distribution Management applications use srs_rules parameters with a SET_NAME of 'PFS' to configure what kind of data sets are used and how the application results are computed and displayed.

To view and edit Power Flow Rules, use the Event Management Rules tab in the Configuration Assistant. Expand the **Power Flow Related Rule** item in the left panel to display the rule categories.

Feeder Load Management History and Accuracy

Historical tracking of Feeder Load Management (FLM) Forecasting is enabled in two steps:

1. An Event Management Rule (SRS Rule) called "ENABLE_FLM_HISTORY" must be activated.
2. Historical logging must also be selected on individual feeders in the Configuration Assistant's "Feeder Management" tab.

Presentation of accuracy results is highly configurable. The FLM Forecast Accuracy page is driven from the FLM_FDR_LOAD_ACCURACY database view. Projects can create this view to display accuracy data in whatever fashion is deemed most appropriate. The product version of this view captures the following:

- Maximum forecast for 3-phase totals of kW, kVAr, True Load, and Generation
- Minimum forecast for 3-phase totals of kW, kVAr, True Load, and Generation
- Average forecast for 3-phase totals of kW, kVAr, True Load, and Generation
- Actual results for 3-phase totals of kW, kVAr, True Load, and Generation
- Accuracy of average forecast for 3-phase totals of kW, kVAr, True Load, and Generation
- Standard deviation of forecasts for 3-phase totals of kW, kVAr, True Load, and Generation

The minimum requirements for the FLM_FDR_LOAD_ACCURACY_VIEW are the following columns:

- FDR_CLS,
- FDR_IDX,
- ANALYSIS_DATE

Note: The system assumes that these columns are present.

The accuracy trend graph can be modified to display any data that is captured in the FLM_FDR_LOAD_ACCURACY_VIEW. Product configuration displays the following:

- Average Forecast of 3-phase Net kW
- Actual value of 3-phase Net kW
- Average Forecast of 3-phase Net kVAr
- Accuracy of 3-phase Net kW
- Accuracy of 3-phase Net kVAr (hidden by default)

Chapter 18

Java Application Configuration

This chapter describes the Oracle Utilities Network Management System Java application framework and methods for creating and deploying customizations. This chapter includes the following topics:

- Understanding the NMS Java Application Configuration Process
- JBot Application Configuration
- NMS JBot Tool Configuration
- Customizing Applications
- JBotCommand Methods Reference

Understanding the NMS Java Application Configuration Process

Oracle Utilities Network Management System Java applications are configured using standard product configuration data with overrides that are customer/project specific. This section describes the product configuration files and explains the process for creating and deploying custom overrides.

This section includes the following topics:

- Understanding NMS Java Application Configuration Files
- Understanding the Process for Building and Deploying Custom Configuration
- Testing the Java Client Configuration

Understanding NMS Java Application Configuration Files

While there is some graphical user interface (GUI) configuration information stored in the database (e.g., certain menu options), most Java application GUI configuration is contained in XML and text files. The primary configuration file types are:

- ***.xml**: the primary Oracle Utilities Network Management System Java application XML configuration files. The ***.xml** files follow the JBot XML schema (jbot.xsd) and must be modified as a whole file to be valid. Most of the attributes in the XML file are either required or have a default value.
- ***.inc**: XML snippets that are referenced in the XML files.
- ***.properties**: standard Java configuration text files. Properties configuration follow a base-plus-delta hierarchy so you only need to include a project version of a properties file when you wish to modify a property and then only need to include the lines that are being modified.

Configuration File Location

NMS Java application configuration files are installed in the `${NMS_BASE}/dist/baseconfig/product/` directory. When creating a custom configuration, the application's configuration files are copied to the `${NMS_CONFIG}/jconfig/` directory using the same directory structure as they exist in the product directory.

For example, the AMR interface configuration file (`AMRInterface.properties`) is located in the `${NMS_BASE}/dist/baseconfig/product/server/` directory. To customize the interface, copy the configuration file to `${NMS_CONFIG}/jconfig/server/` and edit the new file.

Understanding the Process for Building and Deploying Custom Configuration

Once configuration files are modified, the applications must be rebuilt and deployed to the Oracle WebLogic Application Server.

> **Note:** the customizations to the Java application configuration are made and built after the initial installation of the Oracle Utilities Network Management System.

Create or Modify the Project build.properties File

When implementing custom configurations, the project parameters are added to the `build.properties` file. The file contains various properties that are required for the configuration build process.

The file is located in your `${NMS_CONFIG}/jconfig/`directory. If it not in the directory, it needs to be created from a template.

1. Copy `$NMS_BASE/templates/build.properties.template` to `$NMS_CONFIG/[project]/jconfig/.`
2. Rename the template to build.properties.
3. Edit the file to customize it for your environment.

The file includes all of the variables that may be set and their usage. Some of the parameters depend on specific implementation options and may not apply to your project. For example, the config.multispeak_jms parameter is only needed for projects with the MultiSpeak Adapter.

Some of the general values in the build.properties file are:

`project.name`	The name of the project/customer. This is displayed in the Help About dialog of any Java GUI applications to identify the application as being configured for the project/customer.
`project.tag`	This is a version control tag or other identifier used to identify a particular build of the customer-specific configuration. This is also displayed on the Help About dialog of any Java GUI applications to identify a customer-specific configuration deployment.
`publisher.ejb-user`	This is the user that the corba publisher will run as. It should be part of the nms-service group. If not defined, defaults to nms1.
`dir.localization`	If the configuration is based off of a localized (non-English language) version, enter the directory of the localization configuration. Otherwise leave this commented out.

dir.config.deploy	This is the directory where runtime configuration jar files will be created. The default is a staging area ($NMS_HOME/java/deploy), but it is also possible to configure these runtime files to be deployed directly to the application server; if this is desired, uncomment the line and update the path as appropriate for your WebLogic deployment.
dir.working	This is the directory where the configuration files will be copied to.
dir.client_installed	This is the directory where the Oracle Utilities Network Management System Java applications will be installed on a PC.

Build Process for XML and Properties Files

After making changes to the java application configuration files and modifying the build.properties file for your environment, the runtime configuration jar files are created by running the following command:

```
$ nms-install-config --java
```

The command will create the **cesejb.ear** and **nms-multispeak.ear** (*Oracle Utilities Network Management System MultiSpeak Adapter*) files.

If the cesejb.ear file is to be deployed to a staging area, it must be copied to the appropriate directory for the Java application server (*e.g.,* WebLogic).

The build process copies and/or merges the **xml** and **properties** files from the product and project directories to `$NMS_HOME/java/working` and combines them in a jar file.

- When a project **xml** file exists, it takes precedence over the product version.
- Product and project properties files are combined into one generated file.

Example

You want to rename the Work Agenda tool in the Web Workspace **Tools** menu.

Workspace_en_US.properties Product Version:

```
...
# Tools - Options
MNU_TOOLS.text = Tools
MNU_WORK_AGENDA_1.text = Work Agenda 1...
MNU_WORK_AGENDA_1.tooltip = Start Work Agenda 1
MNU_WORK_AGENDA_2.text = Work Agenda 2...
MNU_WORK_AGENDA_2.tooltip = Start Work Agenda 2
MNU_VIEWER_1.text = Viewer 1...
MNU_VIEWER_1.tooltip = Start Viewer 1
...
```

1. Create a new Workspace_en_US.properties file in `${NMS_CONFIG}/jconfig/product/ops/workspace/properties`.
2. Copy the lines that you want to edit from the product version of Workspace_en_US.properties and paste them into the new project file.
3. Edit the lines in the new file:

```
MNU_WORK_AGENDA_1.text = Work List One...
MNU_WORK_AGENDA_1.tooltip = Start Work List One
MNU_WORK_AGENDA_2.text = Work List Two...
MNU_WORK_AGENDA_2.tooltip = Start a second Work List
```

4. Save and close the project file.
5. When you run the build, a generated version of the file will be created that merges the changes into the product properties file:

   Generated Version:

```
# Generated from
projects\jconfig\ops\workspace\properties\Workspace_en_US.properti
es
# $Id: Workspace_en_US.properties,v 1.3 $
MNU_WORK_AGENDA_1.text = Work List One...
MNU_WORK_AGENDA_1.tooltip = Start Work List One
MNU_WORK_AGENDA_2.text = Work List Two...
MNU_WORK_AGENDA_2.tooltip = Start a second Work List

# Generated from
product\jconfig\ops\workspace\properties\Workspace_en_US.propertie
s
MNU_VIEWER_1.text = Viewer 1...
MNU_VIEWER_1.tooltip = Start Viewer 1
…
```

Project values that change a product value will override the product value; however, if a project duplicates, but does not change, the product configuration, then the line is removed from the project configuration in the generated file.

Deploying to WebLogic Application Server

To deploy the Oracle Utilities Network Management System application in your domain, follow these steps:

1. Access the WebLogic Server Administration Console by entering the following URL:

   `http://hostname:port/console`

   Here hostname represents the DNS name or IP address of the Administration Server, and port represents the number of the port on which the Administration Server is listening for requests (port 7001 by default).
2. If you have not already done so, in the Change Center of the Administration Console, click **Lock & Edit**.
3. In the left pane of the Administration Console, select **Deployments**.
4. If a previous release of Oracle Utilities Network Management System (cesejb) is in the table:
   - Select the checkbox to the far left of the deployed cesejb application and click **Stop** and choose **Force Stop Now** to stop the application.
   - Select the checkbox to the far left of the deployed cesejb application. Click the **Delete** button at the top or bottom of the Deployments table to delete the cesejb application, then click Yes to confirm your decision.
5. In the right pane, click **Install**.
6. In the Install Application Assistant, locate the cesejb.ear to install. This will be in the directory listed in your build.properties setting "dir.config.deploy".
7. Click **Next**.
8. Specify that you want to target the installation as an application.
9. Click **Next**.

10. Select the servers and/or cluster to which you want to deploy the application.

    Note: If you have not created additional Managed Servers or clusters, you will not see this assistant page.

11. Click **Next**.
12. Click **Next**.
13. Review the configuration settings you have specified, and click **Finish** to complete the installation.
14. To activate these changes, in the Change Center of the Administration Console, click **Activate Changes**.

Validate Read Only Tables

If running under WebLogic, verify that the nms_readonly.sql accurately defines the tables that the client needs access to. The most likely reason that this list will need to be changed is if a viewer search was added that uses another table or view.

Note that the device attribute tables will always be included so they do not need to be explicitly listed in the readonly.sql file.

If the list needs to be changed from product, copy the nms_readonly.sql to a project specific file and put it in $NMS_CONFIG/sql.

Alternate Deployment of Client-Only Changes

If it is desirable to deploy client-only configuration changes without restarting the managed server each time, the following process can be used:

On the managed server, under startup parameters add this parameter:

```
-Dnms.config_override_dir=/override_dir
```

Replace override_dir with the path to a directory that can contain an updated nms_config.jar file. Ideally when multiple servers for a single instance of nms a shared directory should be used so that each server is updated with one operation.

Then if it is desired to update the existing configuration for a running do the following:

1. Run `nms-install-config --java`
2. Copy `$NMS_HOME/java/working/cesejb.ear/lib/nms_config.jar` to the override_dir defined above.

Then, the next time a client is started, it will use the overridden file.

> **Note:** If this feature is used, it should be done as a process where any update to `cesejb.ear` deletes the obsolete nms_configuration. Otherwise, the managed server will log warnings that there is file older than the one in the cesejb.ear file. In this case, the version that is in the cesejb.ear file will be used.

Most things that are in the `$NMS_CONFIG/jconfig` can be modified this way, with the exception of items in the `$NMS_CONFIG/jconfig/server` directory. Also, most items in the `Global_en_US.properties` and `CentricityTool.properties` files cannot be modified, unless they only affect the client (such as window positions).

Color Configuration

Defining color schemes is defined by modifying ColorSchemes.properties. These define the overrides for colors that are defined in UIManager.properties and themesource.properties. New schemes can be added by adding additional entries in ColorSchemes.properties. All RGB colors are supported.

Testing the Java Client Configuration

This section details how to test Java client configuration without deploying the changes to an app server. Changes can be made locally on a client Microsoft Windows machine and immediately tested.

1. Use the **Configuration Assistant** to create the client application installer and install the application you will be testing (for example, Web Workspace). See the Oracle Utilities Network Management System Installation Guide for complete instructions on creating the installer and installing the client applications.

   Notes:

   - The directions below assume that the client is installed to `C:\OracleNMS` and the project name is [your_project_name]. The location and the project name can be changed as appropriate.
   - **Installer Log Files:** The installers create log files that may be used in troubleshooting installation issues. The log files are saved to:

     `C:\Users\[user]\AppData\Local\Temp\OracleNMS [Windows 7]`

     The log files have the following naming convention: `[applicationname].log` (for example, `WebWorkspace.log`).

2. On the NMS server machine, do the following:

   ```
   $ cd $NMS_CONFIG
   $ zip -r $HOME/nms_config.zip jconfig
   $ cd $NMS_HOME
   $ zip -r $HOME/java.zip java
   ```

3. Next, transfer them to the client machine.

   - Unzip nms_config.zip to `C:\OracleNMS\[your_project_name]`
   - Unzip java.zip to `C:\OracleNMS\`

4. Install Apache Ant version 1.8.2. Be sure to put the ant bin directory on the system path. For example, if Apache Ant is installed to `C:\apache-ant-1.8.2`, add `C:\apache-ant-1.8.2\bin` to the system path.
5. Create two environment variables (using the Microsoft Windows Control Panel):

   - **Name:** `NMS_CONFIG`; **Value:** `C:\OracleNMS\[your_project_name]`
   - **Name:** `NMS_HOME`; **Value:** `C:\OracleNMS`

   You can then modify the configuration in `C:\OracleNMS\<your_project_name>\jconfig`

6. To test the changes do:

```
cd C:\OracleNMS\[your_project_name]\jconfig
ant clean config
```

or

```
ant config
```

Notes:

- **ant clean config** will regenerate all of the configuration; you will need to do that when updating to a new release.
- **ant config** can be used within a session to only update the files that have changed.

7. Finally, run the application as normal. The system will use the local configuration instead of the configuration on the server.
8. When satisfied with the configuration, transfer the files in `C:\OracleNMS\[your_project_name]` to the NMS server under `$NMS_CONFIG`.

JBot Application Configuration

This section includes the following topics:

- JBot Configuration Overview
- Understanding the JBot XML Schema and XML Files
- GUI Configuration
- Advanced Configuration Options
- JBot DataStore Reference

JBot Configuration Overview

JBot is a system developed by the Oracle Utilities Network Management System group for representing GUI forms as XML documents. Product versions of files are stored in `${NMS_BASE}/dist/baseconfig`. Project versions are stored under `${NMS_CONFIG}/jconfig`.

This document contains a description of the configuration needed for all Oracle Utilities Network Management System Java Tools. This includes configuration to:

- Organize all Java Swing Components visually.
- Attach language-independent text and tooltips to the Components.
- Attach specific logic to user actions on the Components.
- Display specific pieces of data held in memory on the Components.
- Set Components' enabled/editable status dependent upon tool-specific States.

Glossary of Terms

Term	Definition
Command	Specific piece of functionality that is executed when a user acts on the GUI.
Component	A member of or enhancement to the standard Java Swing package, including TextFields, TextAreas, Buttons, Tables, Trees, Panels, etc.
Data Store	Collection of data that may be accessed by any *Command* and displayed by any *Component*.
Java	Platform-independent object-oriented computer language.
Properties	Standard Java configuration text file. ***.properties** files define all text and tooltips for each Component.
Swing	Java library of standard visual Components.
Tag	XML key that describes the *Component* to be added. Tags look like this: `<tag_name>`.
Tool	A grouping of Oracle Utilities Network Management System functionality that may be used as an application or an applet.
Tool State	Tool-specific milestones, set as internal flags, that may be used to configure *Components*' enabled and editable statuses. Examples of Tool States: `POPULATED`, `ASSIGNED`, and `CLEARED`.

JBot Component Gallery

This section contains a sample image of each Component, a description of the Component and the Component's name, which is used in the Component's XML tag.

Component Name/ XML Tag	Description
Button	Single clicking on a button will perform a defined Action.
CheckBox	Allows an item to be marked as selected.
CheckBoxMenuItem	A menu item that has a checkbox next to it when it is selected. It is configured just like a MenuItem.
CollapsiblePanel	Collapsible in the horizontal or vertical direction. The purpose is to save screen real estate. The image and the title are configurable.
ComboBox	A list of elements that defaults to showing one element. To select from all of the elements, click on the arrow. The purpose of this Component is to save screen real estate and to only allow the user a finite set of options.
ControlZoneSelector	Popup display of a Control Zone tree, displaying a specified (default 3) # of levels of the control zone hierarchy to allow user selection of a control zone.
CrewIconsPanel	Specialized panel for Crew Actions window.
DateTimeSelector	Pop up (actually more of a dropdown) calendar that allows the user to specify the date/time. It will follow the specified date/ time format set in nms_datefmt.
Label	Text description that is associated with another Component, frequently a TextField.
LabelIndicator	Label whose icon changes with the change in the tool status.
List	Lists can be single or multi-select. The list box will be scrollable when the number of elements exceed the size of the list.
MainPanel, SubPanel	Several Components are placed on a panel to control a section of the GUI. If panels are configured to have visible titles, the system automatically adds a border to delineate the area. Configure panel titles in the properties file as the "text" value, or point the "data_source" attribute to a dynamic Data Store value in the xml file.
Menu	Element of a MenuBar that can have MenuItems, RadioButtonMenuItem, CheckBoxMenuItem, or SubMenu, and Separators (horizontal delimiters).
MenuBar	Bar at the top of a panel that contains one or more Menu elements.
MenuItem	Standard text or icon option in a Menu.
PasswordField	A field that works just as a TextField except that it displays asterisks instead of the characters typed.
PopupMenu	Right-click menu with a number of menu items which when selected performs a defined action.
RadioButtonMenuItem	A choice on a menu that is part of a group where only one can be selected at a time. It is configured just like a MenuItem.

Component Name/ XML Tag	Description
RadioGroup	Similar to a CheckBox, but only one item can be selected at a time.
ScrollPane	It provides a scrollable view of a set of SubPanel Components (which in turn can contain other components). When screen real estate is limited, it is used to display a set of Components that is large or whose size can change dynamically.
Slider	A Component that lets the user enter a numeric value bounded by a minimum and maximum value.
StatusBar	Displays messages to the user. It contains a Oracle Utilities Network Management System icon, and can also have a progress bar and text and label indicators.
SplitPane	Split the two panels by a divider that can be dragged in either direction to increase or decrease the size of each panel.
Table	Data is displayed in a tabular format. They can support single or multi-row selection, and cells can display icons and DateTimeSelectors in addition to dates and strings.
TabbedPane	A component that lets the user switch between a group of components by clicking on a tab with a given title and/or icon. Contains one or more Tabs.
TextArea	Allows the user to enter text on multiple lines. When the number of lines exceeds the viewing area, then the Component is scrollable.
TextField	Allows the user to enter text.
TextIndicator	Changes the displayed text when the tool status changes.
ToggleButton	A two-state button that stays in the pressed position the first time it is clicked. The button returns to the unpressed position the second time it is clicked.
ToolBar	Component below a MenuBar on a panel. It can be automatically generated from the MenuBar by setting <ToolBar use_menu="true"/> . Also contains ToolBarItems and Separators. Use <hide_icon="true"> when you wish the icon to be hidden when the menu item is.
ToolBarItem	Element of a ToolBar, generally with a specified icon.
ToolContainer	Allows a tool to be contained by another tool.
Tree	Data can be presented in a hierarchical order. If a parent has children, then the parent can be opened to display the children or closed to hide them.
TreeTable	A combination of the tree Component and the table Component. This allows a tree to be displayed with multiple columns. Attributes available: **name**="unique component name" **class**="fully qualified class name that overrides com.splwg.oms.jbot.component.JBotPaneTreeTable" See Tree Table XML for sample configuration.
ViewerPanel	Specialized panel used by the Viewer tool.

Understanding the JBot XML Schema and XML Files

Schemas describes the information required to create a valid XML file, what each element has as its child elements, their attributes, and any restrictions on them. The JBot schema has the following conventions:

- **Elements**: every word begins with a capital letter (*e.g.*, **MainPanel**, **SubPanelType**, etc.).
- **Attributes**: every word begins with a lowercase letter; attributes with compound names are separated by underscores (*e.g.*, **name**, **layout_type**, **collapse_direction**, etc.).

JBot Schema

JBot Tool XML files are based on the jbot.xsd schema, which has the following structure:

```
<JbotToolApp>
<GlobalProperties/>
<ToolBehavior/>
<MainPanel>
<MenuBar/>
<ToolBar/>
<PopupMenu/>
<StatusBar/>
</MainPanel>
<BaseProperties>
<Commands/>
<Imports/>
<DataStores/>
<Dialogs/>
<Adapters/>
</BaseProperties>
</JBotToolApp>
```

JBot Element Definitions

GlobalProperties

The GlobalProperties section defines properties that are used for tool specific configuration values. Global properties are often included by reference to a *<toolname>*_GLOBAL_PROPERTIES.inc file.

Example: Workspace.xml:

```
<JBotToolApp ...>
  <JBotTool ...>
    <Include name="WORKSPACE_GLOBAL_PROPERTIES.inc"/>
     .
     .
     .
    </JBotToolApp>
```

The WORKSPACE_GLOBAL_PROPERTIES.inc then defines global properties for the Workspace:

```
<!-- ...
  Modification of this file also requires change and testing of
View Only, Crew Operations and Trouble Maintenance user(s).
  File(s) can be found in:
  ~/src/config/product/jconfig/viewonly/Workspace.xml,
  ~/src/config/product/jconfig/crew_ops/Workspace.xml and
  ~/src/config/product/jconfig/trbl_mnt/Workspace.xml
```

```
-->
<!-- Used in Workspace.xml -->

<GlobalProperties>
  <StringProperty name="product_name" value="CREW"/>
  <StringProperty name="startLoginTools" value="WorkAgendaTool,
CrewIcons, EventDetails, ControlTool, TroubleInfoTool, CrewInfo,
FaultLocationAnalysisTool, FLMTool, ViewerPanelTool, DDSAlarms"/>
  <StringProperty name="displayLoginTools" value="WorkAgendaTool,
AuthorityTool, CrewIcons"/>
  <IntegerProperty name="work_space.max_viewers" value="2"/>
</GlobalProperties>
```

ToolBehavior

Typically defines what commands to run upon opening or closing the dialog.

MainPanel

Defines the GUI layout of the tool. Includes the following elements:

- MenuBar
- ToolBar
- PopupMenu
- StatusBar

BaseProperties

Contains the configuration that matches JBot names with Java classes.

Commands

This section defines a command. If a command is used either by the tool or by a dialog called from the tool, it must be listed here.

It is preferable to refer to **Commands** using the (class) **name** attribute rather than define the name in a child **CommandClass** element.

For example, the following:

```
<Commands name="CMD_FOO"/>
  ...
<CommandClass name="CMD_FOO"
 class="com.splwg.oms.client.workagenda.FooCommand"/>
```

is equivalent to:

```
<Command name="com.splwg.oms.client.workagenda.FooCommand"/>
```

However, if there is an **Import** section, the system will attempt to find the command(s) in each package. Thus, the following:

```
<Command name="com.splwg.oms.client.workagenda.FooCommand"/>
```

becomes:

```
<Command name="FooCommand">
...
<Imports>
```

```
    <Import name="com.splwg.oms.client.workagenda"/>
  </Imports>
```

Imports

This section defines paths for commands so that a command can be used without specifying the full path.

Datastores

All datastores that are used by the tool or any dialogs called by this tool must be listed here. However, a tool is allowed to use a datastore defined by a different tool, as long as the other tool is loaded first. There are also some instances where a datastore can be defined in the code. This is mainly the case in the crew tools.

Dialogs

Dialogs in the list will be loaded on tool startup; dialogs that are not in the list will be loaded the first time they are called.

Adapters

This section is no longer necessary. If an existing JBot configuration file has this section, it can be removed without a problem. If such a tag does exist, it is ignored.

Include Elements

Runtime Include Elements

use standard XML based xi:include tags. The included files are delivered to the client and they are combined by the application at runtime. This allows for specific XML code that is repeated to be defined once, but used in multiple places.

To define an include file, xmlns, xmlns:xsi, and xsi:schemaLocation must be defined.

For example given this XML fragment:

```
  <Perform name="HLM" category="onMessage"
  type="APPLY_SAFETY_FILTERS">
```

should be changed to:

```
<Perform name="HLM" category="onMessage" type="APPLY_SAFETY_FILTERS"
    xmlns="http://www.ces.com"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.ces.com http://localhost/xml/
jbot.xsd">
    XML code that will be used by multiple tools
    ...
</Perform>
```

The include files should be saved with an `.xml` extension.

To reference this file in another XML file, use the following syntax:

```
    <xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
    href="/SafetyStartup.xml" parse="xml"/>
    ...
```

This allows the second XML document to use the XML code defined in the include file. The above example defines filters that will be used by multiple tools within Web Switching. Therefore, when filters need to be changed, they can be changed once and it will be applied to all tools that are using the **include** file.

Build Time Include Elements

the main limitation on xi:include tags is that they can only be used to insert a single element. While that approach works fine in the body of a JBot configuration, it doesn't work well for inserting tool properties, actions, or datastores.

In these cases, it is easier to use the build time based <Include> element. In this case the build process that creates the nms_config.jar file will perform the inclusions.

These include files should be saved without any extra attributes, and saved with an .inc extension.

```
<Perform name="HLM" category="onMessage"
type="APPLY_SAFETY_FILTERS">
...
```

To reference the file, use the following syntax:

```
<Include name="SafetyStartup.inc"/> s
```

JBot Commands

JBot commands are operations performed as a result of an event. Some examples of events are button presses, table editing and row selecting. Commands are defined in a "Perform" tag. The actual options for the Perform tags vary with the component type.

Here is an example of configuring a command to be run when a menu is selected:

```
<MenuItem name="MNU_EMERGENCY_CONTENT_SELECTION"
 icon="Preferences16.gif">
  <PressPerform>
    <Command value="DisplayDialogCommand">
      <Config name="dialog" value="DLG_EMERGENCY_CONTENT_SELECTION"/>
    </Command>
  </PressPerform>
</MenuItem>
```

This will display the dialog `DLG_EMERGENCY_CONTENT_SELECTION`.

JBot Command Reference

There are many JBot commands available. The HTML-based command reference is available at:

```
$NMS_BASE/documentation/command_doc/index.html
```

All commands accept the following `<Config>` parameters:

- **runInTool**: the JBotTool that this command should run in. This defaults to the current tool.
- **abortable**: whether this command should be aborted when a previous command aborts. This defaults to *`true`*, but can be configured as *`false`* in the rare case that there is a `Command` that should be executed, say, even when a dialog is canceled that sets the abort flag.

JBot Actions

JBot actions allow you to define a list of commands that can be reused multiple times in a configuration. Defining a command directly on a component works well if there is only one place where the command is needed. However, if there are multiple places where the same commands are called, such as a menu item and a button, this provides a way to only define the action once.

Actions should be defined in the ToolBehavior or DialogBehavior tags.

```
<Action name="ACT_PRINT">
   <Command value="DisplayDialogCommand">
      <Config name="dialog"
      value="DLG_PLANNED_REPORT_CONTENT_SELECTION"/>
   </Command>
   <Command value="GenerateReportCommand" when="GENERATE_REPORT">
      <Config name="report_location" value="/Webswitching/
      PlannedSwitching/PlannedSwitching.xdo"/>
      <Config name="parameter_datastore"
      value="DS_PLANNED_REPORT_CONTENT"/>
      <Config name="base_file_name" value="SwitchPlan"/>
      <Config name="file_description" value="report"/>
      <Config name="show_progress_dialog" value="true"/>
      <Config name="dest_file_reference" value="REPORT_FILE_REF"/>
      <Config name="dest_datastore"
      value="DS_PLANNED_REPORT_CONTENT"/>
   </Command>
</Action>
```

Then to use this action, the `ExecuteActionCommand` command should be called:

```
<MenuItem name="MNU_PLANNED_PRINT" icon="Print16.gif"
accelerator="control P">
   <PressPerform>
      <Command value="ExecuteActionCommand">
        <Config name="action" value="ACT_PRINT"/>
      </Command>
   </PressPerform>
</MenuItem>
```

The action is run just like another jbot command. Other commands or actions can also be defined before or after the action command, just like any other jbot command.

Actions can also call other actions, by using the `ExecuteActionCommand` from within another action.

It is also possible for actions to take parameters. For example:

```
<Action name="ACT_GGENERATE">
   <Command value="GenerateReportCommand" when="GENERATE_REPORT">
      <Config name="report_location" value="/Webswitching/
      PlannedSwitching/$REPORT_NAME$.xdo"/>
      <Config name="parameter_datastore"
      value="DS_PLANNED_REPORT_CONTENT"/>
      <Config name="base_file_name" value="SwitchPlan"/>
      <Config name="file_description" value="report"/>
      <Config name="show_progress_dialog" value="true"/>
      <Config name="dest_file_reference" value="REPORT_FILE_REF"/>
      <Config name="dest_datastore"
      value="DS_PLANNED_REPORT_CONTENT"/>
   </Command>
</Action>
.
.
.
```

```
<MenuItem name="MNU_PLANNED_PRINT" icon="Print16.gif"
accelerator="control P">
    <PressPerform>
        <Command value="ExecuteActionCommand">
          <Config name="action" value="ACT_PRINT"/>
          <Config name="$REPORT_NAME$" value="PlannedSwitching"/>
        </Command>
    </PressPerform>
</MenuItem>
```

This will replace the token **$REPORT_NAME$** with the value PlannedSwitching. Any text in the configuration can be replaced this way. You cannot, however, replace the Command names themselves.

If you wish to use an **Action** defined in a different XML file there are two options. The first option is if you wish to run the action in the other tool. In that case, you can use the "runInTool" option, like other commands. However, if you wish to run the action in the current tool, even though it is defined in another tool, use the **tool** config option.

When Attributes

Various elements can have a **when** clause that indicates that the configuration should only be used if the **when** clause returns **true**. For example:

```
<Command value="SetDataStoreValuesCommand"
    when="VIEWER_STUDY_MODE and (DS_FLM_GLOBAL.PF_IS_ON=='true')">
    <Config name="sources" value="'JAVA_VIEWER', 'PF'"/>
    <Config name="targets"
    value="DS_FLM_GLOBAL.STARTED_FROM, DS_FLM_GLOBAL.STUDY_FOR"/>
</Command>
```

This will only be run if the status flag `VIEWER_STUDY_MODE` is **true**, and if the datastore value `DS_FLM_GLOBAL.PF_IS_ON` equals the text literal **"true"**.

The following example will only show the component if the datastore value is **not Y**.

```
<Visible initial="true" when="$DS_ABNORMAL_CONDS.PRI_TEXT != 'Y'"/>
```

Note that the `$` in front of the system is used to indicate the value is a string, if it has not been populated yet. Otherwise this would not return a true because a comparison with a null always returns `false`.

Validation Toolkit

The validation toolkit provides a way of validating user data, as well as another way of coloring tables or other components. While most JBot commands work off of the underlying data, the validation toolkit works off of GUI widgets.

The **Validation Testing Tool** shows various examples what can be configured using the validation toolkit. The example is in:

`jconfig/ops/test/xml/Validation.xml.`

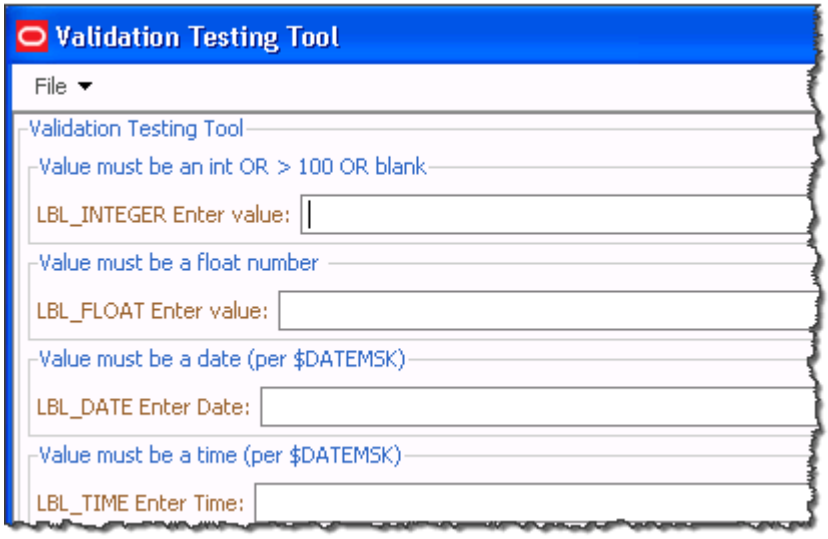
To run this tool, add the following code to `WorkspaceMenuBarTool.xml` to create a menu item that will start the Validation Testing Tool:

```
<MenuItem name="MNUITM_VALIDATION" hide_icon="true">
    <PressPerform>
        <Command value="DisplayToolCommand">
            <Config name="tool" value="Validation"/>
            <Config name="class" value="com.splwg.oms.jbot.JBotTool"/>
        </Command>
    </PressPerform>
</MenuItem>
```

In `WorkspaceMenuBarTool_en_US.properties` add:

```
MNUITM_VALIDATION.text = Demo Tool...
```

When you click on the new menu choice, it will display the tool:



Feel free to try various values in the form to understand how the validation toolkit can be used. The form is defined by **Validation.xml**. To use this form, find the section on the from that is like the validation you wish to perform, then look at `Validation.xml` to see the code needed.

Locale-Specific Properties File

Property Naming Convention

The `[toolname]_[language]_[locale].properties` file contains all language related properties for the components. They are identified with the syntax:

```
KEY.property = value string
```

Property	Component Type	Description

Text	All Actions	String displayed on the Button that invokes the Action.
		
		Syntax:
		`ACT_KEY.text = string`
		
		Example:
		`ACT_SET_START_DATE.text = set starting date`

text Radio	Button group members, table headers, combo box entries	String displayed on the piece of a larger component.
		
		Syntax:
		`RBG_KEY.ENTRY_KEY.text = string`
		
		Example:
		`RBG_OUTAGE_TABLE.H_IDX.text = Device index`

tooltip	All Components and Actions	Tool tip string.
		
		Syntax:
		`KEY.tooltip = string`
		
		Example:
		`ACT_SET_START_DATE.tooltip = Set the starting date to the current date`

When there is no locale, the tool tries these file names.

1. `[toolname].properties`
2. `[toolname]_en_US.properties`

If there is a locale defined, the tools will try these file names.

1. `[toolname]_[language]_[locale].properties`
2. `[toolname]_[language].properties`
3. `[toolname].properties`

Reserved Words

Here are some reserved words that you may use in property files. It is recommended that when used, it should be placed at the top of the file.

reserved word	What it does
include	List of additional property files to read in. This list must be separated by spaces. If you want the current file to override the included property file, use propertyname_SELF.
includeList	Returns the list of property files that were read in. This value is set by the PropertyReader class and cannot be overridden.
includeListCount	Number of property files read in. This value is set by the PropertyReaderclass and cannot be overridden.

Sample use of include is in MessageCode_en_US.properties file.

```
# List the other files to include as part of reading in
# this property file. Just the base name is needed.
# Must be space delimited only!
include = CoreResources
```

The `include` directive should be treated like a "replace or add" operation. Any property found in the current file that is also in the included file will be replaced by what is specified in the included file. In order to override settings in an included file, you need to add an additional property file with those overridden values or you can specify `[base property name]_SELF` to use the properties in the current property file. Specifying this "_SELF" option as the last included file will ensure that the settings in this file will always take precedence over any included property value.

Example

This example demonstrates usage of `include` where we override some properties from the included property file.

`SwmanTool_en_US.properites` has many property settings including the following entry:

```
TAB_STEPS.text=Steps
```

`SwmanEmergencyTool_en_US.properties` may have very few settings and may only have the following:

```
include = SwmanTool SwmanEmergencyTool_SELF
# Override label for Steps tab
TAB_STEPS.text=Emergency Steps
```

In this case, all of the properties in `SwmanTool_en_US.properties` will be included in the `SwmanEmergencyTool_en_US.properties` property file along with the one overridden property setting for `TAB_STEPS.text`. Since ordering matters when including properties, the properties in the current file (`SwmanEmergencyTool_SELF`) will override the properties from the previously included files.

If no properties need to be overridden, then the inclusion of include file [base property file name]_SELF or in the example above "`SwmanEmergencyTool_SELF`" is not required.

GUI Configuration

Laying Out Components

Layout values are based on Java's GridBagLayout component.

Modify Fill

The fill value is a string. When set to *BOTH*, the component will fill its entire x,y coordinates. When set to *NONE*, the component will fit only the area that it needs to. For example, if a button is set to *NONE*, then the button will only fill around the text. To be even more specific, if two text letters are on a button, then it will be smaller than if there are six text letters on the button.

Fill can also be specified to be *HORIZONTAL* or *VERTICAL* for specific fill in one direction. Note that for labels, fill should generally be set to *NONE*. If it is not *NONE*, then attempts to right-justify the label by setting the anchor to "*EAST*" will fail.

Modify Insets

Insets are given as four different values: top, bottom, left, and right. Each of these values will buffer a component from all other components. For example, if all of the values are 2, then the component will be two pixels on all four sides from the closest components.

Modify Weight

The weight is given as x and y values. The x stands for horizontal and y stands for vertical. The weight indicates how much to stretch the component relative to the other components on the frame.

Choosing the Font

Labels can have their font defined by the optional <Font> tag under the <LabelBehavior> tag.

```
<LabelBehavior>
  <Font name="Tahoma-BOLD-24"/>
</LabelBehavior>
```

Oracle Utilities Network Management System code uses the Java Font.decode() method (see the Java Font class documentation for further information).

To ensure that this method returns the desired Font, format the name parameter in one of these ways:

- fontname-style-pointsize
- fontname-pointsize
- fontname-style
- fontname
- fontname style pointsize
- fontname pointsize
- fontname style
- fontname

in which style is one of the four case-insensitive strings: "PLAIN", "BOLD", "BOLDITALIC", or "ITALIC", and pointsize is a positive decimal integer representation of the point size. For

example, if you want a font that is Arial, bold, with a point size of 18, you would call this method with: "Arial-BOLD-18".

If a style name field is not one of the valid style strings, it is interpreted as part of the font name, and the default style is used.

Only one of ' ' or '-' may be used to separate fields in the input. The identified separator is the one closest to the end of the string that separates a valid pointsize or a valid style name from the rest of the string. Null (empty) pointsize and style fields are treated as valid fields with the default value for that field.

Some font names may include the separator characters ' ' or '-'. If str is not formed with three components (*e.g.*, style or pointsize fields are not present in str) and fontname contains the separator character, then these characters may be interpreted as separators. In this case, the font name may not be properly recognised.

The default size is 12 and the default style is PLAIN. If the name does not specify a valid size, the returned Font has a size of 12. If the name does not specify a valid style, the returned Font has a style of PLAIN. If you do not specify a valid font name in the name argument, this method will return a font with the family name "Dialog".

Bold and Italic Labels

Labels can be defined as plain, bold, italic, or bold italic. This is done by the optional <Font> tag under the <LabelBehavior> tag.

This is an example of an italic label:

```
<Label name="LBL_ITALIC_TEXT">
  <LabelPlacement start="0,relative"/>
  <LabelBehavior>
    <Font style="ITALIC"/>
  </LabelBehavior>
</Label>
```

This is an example of a bold label:

```
<Label name="LBL_BOLD_TEXT">
  <LabelPlacement start="0,relative"/>
  <LabelBehavior>
    <Font style="BOLD"/>
  </LabelBehavior>
</Label>
```

This is an example of a label that is neither bold or italic:

```
<Label name="LBL_NORMAL_TEXT">
  <LabelPlacement start="0,relative"/>
</Label>
```

This is an example of a label that is both bold and italic:

```
<Label name="LBL_BOLD_ITALIC_TEXT">
  <LabelPlacement start="0,relative"/>
  <LabelBehavior>
    <Font style="BOLD ITALIC"/>
  </LabelBehavior>
</Label>
```

Advanced Configuration Options

This section describes components that provide more intricate configuration options.

JTable

1. **Column Editor** - A column in a table can be specified to have a different component for editing its cells. The valid components that can be specified are a ComboBox, a CheckBox, a TextField, or a TableCellTextArea. When a column has a different editor, such as a ComboBox, then all the rows in the table have a ComboBox for that column. A specific editor, rather than the default one, is generally specified when we want that column to be editable. When an editor is specified for a column, we should make sure that we provide all the necessary configuration options for that editor.

2. **Column and Row Popup Menus** - This option specifies the name of the right click pop up menus, which would show up when a user right clicks on a column header or one of the rows of the table. The name should be a valid name as per the name of the pop up menus that are already created while parsing the XML file.

3. **Status Keys** - The background and the foreground color of the rows in the table is configurable as per the contents of that row. The list of all the possible statuses for which we want the background and foreground colors to change are provided by status keys. The status keys are specific to the table and they should be valid values in a column of the data store from which the table obtains its data. This column is configured as the `status_column` in the `<TableBehavior>` element. Note that these row colors will be used to color the first visible column of the row if the user enables "Cell Coloring" instead of "Row Coloring" and no Cell Colors are configured for this table. All RGB colors are supported.

4. **Column Visibility** - the Column element allows a Visible sub-element with attributes for "initial" and "when," which behave like the Visible elements available for other Components.

5. **Column Justification** - In tables, text is typically left justified and numbers are typically right justified. It is possible to override the justification on a per-column basis by using the justification attribute: `<Column key="EVENT_IDX" justification="`*left*`"/>`

   The options are:

   - `left`: the column is left justified.
   - `right`: the column is right justified.
   - `center`: the column is center justified.
   - `general`: numbers are right justified and other data is left justified. (Default)

6. **Text Wrapping** - To wrap text in a column, set the WrapText element to true:

```
<Column key="swmanStep.operationOutcome">
  <Editable initial="true"/>
  <WrapText initial="true"/>
  <Editor>
    <TableCellTextArea/>
  </Editor>
</Column>
```

To make a wrapped column editable, use the TableCellTextArea editor:

```
<Column key="swmanStep.operationOutcome">
  <Editable initial="true"/>
  <WrapText initial="true"/>
  <Editor>
    <TableCellTextArea/>
  </Editor>
</Column>
```

7. **Preferred Column Widths** - To set columns within an auto-resized table to use a preferred column width, a minimum and max column width will need to be specified. Thus, the column can be resized within the limits of the minimum and maximum setting. When the table is initially displayed, it uses the preferred size, which is the existing "width" property setting.

   Example:

   XML - Table Behavior Definition

   ```
   <TableBehavior auto_resize_columns="true"
   data_source="DS_EXAMPLE">
     <Column key="Idx" />
     <Column key="Cls" />
     <Column key="Description" />
   </TableBehavior>
   ```

   Property - Table Column Settings

   ```
   TBL_EXAMPLE.Idx.text=Number
   TBL_EXAMPLE.Idx.minimum_width=10
   TBL_EXAMPLE.Idx.width=90
   TBL_EXAMPLE.Idx.maximum_width=150

   TBL_EXAMPLT.Cls.text=Type
   TBL_EXAMPLT.Cls.minimum_width=10
   TBL_EXAMPLT.Cls.width=90
   TBL_EXAMPLT.Cls.maximum_width=150

   TBL_EXAMPLT.Description.text=Description
   ```

   In this case, the table will be initially drawn with the first two columns having a width of "90" and the Description column spanning to utilize the rest of the space given to the JTable component. The first two columns can be resized, but only down to a width of 10 and up to 150. If the entire table is squished, the Description column will be cut down until all the columns have reached their preferred width. At which point all the columns will be squished at the same rate. Since the Description column does not have a preferred width, the width of the label ("Description") is used.

8. **Defining Column Headers** - A column header can be defined as either text or an icon. See the following example:

   ```
   TBL_WA_ALARMS.STATUS.text = Status
   TBL_WA_ALARMS.STATUS.icon = status.png
   TBL_WA_ALARMS.STATUS.tooltip = Event Status
   ```

   The image file specified for an **icon** should exist in the tool's images configuration directory, along with all other image files.

   Define a **text** value for all column headers, including those defined as icons. The **text** value will be used in various dialogs where the column name is displayed.

The **tooltip** is used to define a message that will pop up when the mouse is hovered over a column header. If an **icon** is defined, and a tooltip is not, the system will automatically use the **text** value as the tooltip.

9. **Defining Cell Attributes** - If you do not wish to color your entire row, you can color individual cells using the `<CellColor>` element. The `check_column` is the column whose value will be compared, and the "key" will be the column whose color is changed. These cell colors are used both when Row Coloring is enabled and when Cell Coloring is enabled. All RGB colors are supported.

```
<CellColor check_column="FLISR_STATUS" key="EVENT_IDX">
    <Status key="-1" bg_color="yellow" fg_color="black"/>
    <Status key="1" bg_color="green" fg_color="black"/>
    <Status key="2" bg_color="pink" fg_color="black"/>
    <Status key="REDLINED" fg_color="black" bg_color="red"
     font="strikethrough"/>
</CellColor>
```

   The `font` attribute can be a particular font name or an attribute, such as strikethrough, bold, italics, subscript, or superscript.

10. **Defining Cell Separators** - If you wish to have table cell separators, add the following to the corresponding TableBehavior elements:

```
show_horizontal_lines="true"
show_vertical_lines="true"
```

11. **Changing Cell Separator Color** - The default table cell separator color is a light gray. To change the color, add a line to `$NMS_CONFIG/jconfig/global/properties/theme.properties`:

    `table.normal.gridcolor=<rgb hex value>`

    For example, to change the table cell separator to black (hex #000000):

    `table.normal.gridcolor=#000000`

12. **Adding Tiebreaker Sort Field**.- If a sort contains duplicate values, then there is no guarantee that the sort will always display in the same order. Setting a tiebreaker column will use the indicated column as the last column of any sort to ensure that the order remains consistent.

13. Defining Lock Columns - Add the last column to be locked as a lock_column attribute to the TableBehavior tag. For example:

    `lock_column="FEEDER_ALIAS"`

    Note that locked columns cannot have a Visible "when" attribute configured for them.

14. **Changing the Cell Coloring When Selected** - Normally, the entire row is shown as the selected color when it is selected on. Adding the TableBehavior attribute show_cell_color_on_select="true" will show the cell color even when the row is selected.

For other properties that can be modified, see the product version of theme.properties.

Drag and Drop Support

Tables, Tree Tables, Crew Icons, and Crews in the Viewer can be configured to respond to drag and drop. Using the same mechanism, copy and paste can also be supported.

To support Dragging from a component, a `DragSource` should be configured in the behavior tag. See this example from Workagenda.xml:

```
<DragSource enable_cut="false" enable_copy="true" enable_drag="true"
flavor="EVENT"/>

<DragSource enable_cut="false" enable_copy="true" enable_drag="true"
flavor="String:"/>
```

This enables dragging and copying. There are two defined "flavors." The EVENT means that it can be the source of a component has a DropPerform configured to EVENT. The `"String:"` defines a comma separated format that can be copied to the clipboard and saved to other applications, such as Microsoft Excel.

By default, all columns will be copied to the clipboard. If only a subset is required, they may be defined by listing them in the flavor:

```
<DragSource enable_cut="false" enable_drag="true"
flavor="String:crewInfo.effectiveZoneName,crewInfo.crewId,crewInfo.cre
wType,crewInfo.contact,crewInfo.crewSize,crewDispatch.feeder_alias,cre
wDispatch.working_at_alias,estRestoreDateTime,crewInfo.numberOfAssignm
ents,hoursWorked,#MOBILE_TYPE,#SHIFT_STATUS,#JOB_STATUS"/>
```

Drag support can be enabled or disabled by adding a "when" attribute. For example:

```
<DragSource enable_cut="false" enable_copy="true" when="!VIEW_ONLY"
enable_drag="true" flavor="EVENT"/>
```

DropPerform

DropPerform should be defined as part of the elements behavior element.

```
<DropPerform enable_drop="true" enable_paste="true" flavor="CREW_KEY">
  <Command value="com.splwg.oms.client.crew.AssignCrewCommand"/>
</DropPerform>
```

This example supports both dropping and pasting to component. It will accept any components with a DragSource flavor of `CREW_KEY`.

Dialog Configuration Options

JBot dialogs are defined in xml files that start with DLG_. They have a similar configuration to JBot tools. One difference is that the dialogs use the datastores and jbot statuses of the tool they are attached to.

Here is an example of the start of a dialog configuration:

```
<JBotToolDialog width="280" height="120" modal="false"
  always_on_top="true">
```

- **width**: The width of the dialog.
- **height**: The height of the dialog.

  Note: If the height and width are not defined, it will use the natural size of the components that are initially visible.
- **modal**: If the dialog requires the dialog be dismissed before using another part of the system, then this should be set to true. If the dialog only needs to be dismissed before using that particular tool, then this should be set to true; otherwise, it should be set to false.

- **always_on_top**: If a dialog should remain on top even if another window is selected, set this to true. Note that modal dialogs are implicitly always on top and do not need this attribute specified.

Performing Actions When Tools and Dialogs Open or Close

If a command or a list of commands needs to be run in response to a window action, such as a tool opening or closing, it can be defined using the `<ToolBehavior>` and `<DialogBehavior>` tags. These tags use a `<Perform>` subtag that takes a name and a category. The "name" attribute should be "Window" and the category name will be either `windowOpened` or `windowClosing`. `windowOpened` will allow the users to run code when the window opens for the first time. `windowClosing` will run when the users has requested that the tool close, but before the system actually closes the window (to allow the system to validate data, etc.). Other window events can also be caught. Please see the Java documentation for the WindowListener Interface for further information; the methods in that class can be used as the "category" attribute in this tag.

Here is an example on running a command when a tool opens:

```
<ToolBehavior>
   <Perform name="Window" category="windowOpened">
      <Command value="DoSomethingCommand"/>
   </Perform>
</ToolBehavior>
```

Here is an example of a command running when a tool closes:

```
<ToolBehavior>
   <Perform name="Window" category="windowClosing">
      <Command value="QuitCommand"/>
   </Perform>
</ToolBehavior>
```

Setting component height and widths normally, the size of the tool, along with the weight and fill attributes determine the size of the components. However, sometimes it is necessary to have a component be a certain size. To do this, specify a component_width and component_height attributes in the behavior tag. For example:

```
<Table name="TBL_WA_SUMMARY">
   <TablePlacement start="0,0" width="8" height="1" weight="1,0"
   fill="HORIZONTAL" insets="2,2,2,2" anchor="NORTHWEST"/>
   <TableBehavior data_source="DS_WA_SUMMARY" resize_columns="true"
   auto_resize_columns="false" component_height="59">
```

Calculated Fields

JBot has a rather complicated way of defining text substitution and formatting of fields. Normally, a component refers to a column as it exists in a datastore. (For example, DS_TABLE.code refers to the column code in the datastore DS_TABLE.) This section has examples of most of the different combinations that can be done with calculated fields. For each example, the field name and a sample output is given. The output uses the following datastore as its source:

Status	Priority	Code	Date
A	1	N	1/2/08 12:33
B	4	O	1/4/07 3:33
C	10		1/4/07 3:33
D	20	Q	1/4/07 3:33

Calculated fields are indicated by preceding the field with a #. The format is:

```
#field1, field2, …%[format definition]
```

The format definition is based on the java.text.MessageFormat class. (Please refer to the official Java documentation for more information on the MessageFormat class.)

Examples of Calculated Fields

Concatenating two fields together with a comma separating them:

```
#Status,Code%{0},{1}
A,N
B,O
C,
D,Q
```

Concatenating two or more fields together with a space separating them:

```
#Status,Code%{0} {1}
A N
B O
C
D Q
```

Replacing a field's value with another value:

```
#Status%{0}||A|Status 1|B|Status B
Status 1
Status B
C
D
```

> **Note**: If a value isn't defined, then the original value is used unless the default value is provided. In the example above, if the value of the Status field is 'A' then it is replaced with 'Status 1' and if the value is 'B' then it is replaced with 'Status B'. Otherwise, it is unchanged. The default value is configured by adding a single value to the end of the list (`#Status%{0}||A|Status 1|B|Status B|Default`).

Replacing a field's value with a value from a property file:

Add the following lines to the JBotFormat_en_US.properties files:

```
STATUS.A = Status 1
STATUS.B = Status B
```

Then follow this example:

```
#Status%{0}|||STATUS
Status 1
Status B
C
D
```

> **Note:** Default value can be configured by adding entry, which does not have the original value, to the property file. In the example above such an entry would be: `STATUS = Default Status`.

Replacing an integer code with a string :

```
#Priority%{0,choice,1#Priority A|5#Priority B|10# Priority C}
1, 4, 10, 20
Priority A, Priority B, Priority C, Priority C
```

Note that if the value is greater than the last choice, it will use the last choice. Likewise if a value is less than the first value, it will use the first value. Otherwise, it will use the largest lookup value that is not greater than the original value.

Performing a conditional:

```
#Status,Code,Priority %{0=B?1:2}
1
O
10
20
```

Performing a conditional if a value is null:

```
# Code,Status%{0=null?1:0}
N
O
C
Q
```

Displaying Date and Time Fields

Date, time, and date/time fields use the formatting defined in the Global_en_US.properties file. The following table includes the product configuration as an example.

Format	Value	Keyword	Example Setting
Date%{0}	01/02/08 12:33	Centricity.DateTimeFormat	MM/dd/yy HH:mm
Date%{0,date}	01/02/08	Centricity.DateFormat	MM/dd/yy
Date%{0,date,long}	01/02/08 12:33	Centricity.LongDateFormat	EEE, MMM dd HH:mm
Date%{0,time}	12:33	Centricity.TimeFormat	HH:mm
Date%{0,time,long}	12:33:30	Centricity.LongTimeFormat	HH:mm:ss

Date/Time Custom Sub-Formats

In addition, five custom date formats are available to configure in the Global_en_US.properties file.

- Centricity.SubFormat1= MM/dd/yy HH:mm:ss.SSS
- Centricity.SubFormat2=
- Centricity.SubFormat3=
- Centricity.SubFormat4=
- Centricity.SubFormat5=

This allows for the definition of additional date/time formats. For example, displaying time with milliseconds precision:

Format: Date%{0,date,subformat1}

Value: 01/02/08 12:33:44.123

Returning a value if a substring matches

To define coloring rules, it is sometimes helpful to return a value if a value matches anywhere in a column. This can be done with the MatchSubstringFormat. The following example will return "Out" if "Out" appears anywhere in the string, or an empty string if it does not contain it.

```
<Column name="#TROUBLE_OUT"
definition="#demo.MatchSubstringFormat(Out,TROUBLE_CODE)%{0}"/>
```

Using regular expressions to parse values

It is possible to use regular expressions to parse data. For example, to create a column that displays the area code of a phone number (where the phone # is in the format `###-###-####`), the following can be used:

```
<Column name="#AREA_CODE" definition="#
MatcherFormat(...).*,PHONE_NUMBER)%{1}"/>
```

To remove the dashes from a phone number:

```
<Column name="#PHONE" definition="# MatcherFormat((.*)-(.*)-
(.*),PHONE_NUMBER)%{1}{2}{3}"/>
```

If you have a task that could be done with either MatchSubstringFormat or MatcherFormat, it is best to use MatcherFormat as it is more efficient

For additional information on regular expressions syntax, see:

https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html

How many percent characters (%) do I need?

A percent character (%) defines the start of a format.

A single % means that the original value should be used for both equality testing (such as cell filtering) and sorting. In other words if two source values are mapped to the same display value, then the underlying value will be used for things like cell filtering. If there is a unique mapping, however, performance is the best with this option.

A double (%%) means that the formatted value should be used for equality testing, but for sorting purposes the underlying value should be used. This would be appropriate for priority text strings. For example, if you had a priority field that got mapped to "Emergency", "Priority", "Routine", and "Planned," it would allow the sorting based underlying code instead of alphabetically.

A triple (%%%) means that the formatted value should be used both for equality testing and for sorting.

Status Flags

When dealing with tools that have a parent-child relationship, status flag values are checked in the parent tool unless the status flag is preceded with a percent character (%). This format is used when checking the value of a status flag in a Visible-when or Enabled-when clause within an XML configuration file. For example, a parent-child relationship exists between switching sheets and it's associated safety documents. The safety documents are considered to be child tools of the switching sheet. To check a status flag in the child tool that is used in both the child and parent, then use the % in front of the status flag name.

For example:

```
<Enabled initial="false" when="%TBL_TAG_DEVICES_SELECTED"/>
```

JBot DataStore Reference

To aid the implementer in determining the available columns for a **datastore**, it is possible to create a report that contains all the current values of a **datastore** for a running system. Because each system can have different configured columns, it is necessary to create this documentation from a running system.

Creating the JBot DataStore Report

1. Start the Java application you wish to document. Ensure that the tools you are interested in are populated.
2. Bring up a shell window as the nms user on the nms server.
3. Type:

   ```
   Action any.publisher* ejb client <client id> jbot_report c:/
   OracleNMS/datastore_report.txt
   ```

   This will create the datastore report on the specified client machine. If you wish to change the location that the report will be stored, change the above command.

Reading the Datastore Report

The report contains all the **datastore**s that are currently valid for the application, along with all of the valid columns.

See the following excerpt from the report, which describes the DS_WA_ALARMS **datastore** from Work Agenda.

```
Datastore: :DS_WA_ALARMS
  CUSTOMER_NAME=
  COND_PHASES=B
  COND_STATUS_NAME=RDO
  EST_REST_TIME=07/27/09 14:02
  DEVICE_ALIAS=xfm_oh_JO-9976
  CTRL_ZONE_NAME_5=JO CO 9362
  CTRL_ZONE_NAME_6=
  CRIT_K=1
  ...
```

Note that some of the columns listed are objects that would never be printed. For example, see the excerpt below:

```
  crew.zone_hdl=com.ces.corba.CES.Handle@1646de5
  crew.zone_hdl.class_number=4802
  crew.zone_hdl.app=0
  crew.zone_hdl.instance_number=1001094
```

The `crew.zone_hdl` is an object that would not be displayed. That object contains the class_number, instance_number, and app, which can be displayed.

NMS JBot Tool Configuration

The `${NMS_BASE}/dist/baseconfig/product/ops` folder contains sub-folders with each tool's configuration information. Tool folders typically contain the following sub-folders:

- **images**: contains images used by the tool
- **properties**: contains Java **.properties** files
- **xml**: contains **.xml** and **.inc** files

This section includes the following topics:

- User Permissions
- User Type Configuration
- Login Tool Configuration
- Master Window Configuration (Oracle Fusion Client Platform)
- Work Agenda Configuration
- Crew Actions Configuration
- Event Details Configuration
- Viewer Configuration
- Feeder Load Management Configuration
- Model Management Application Configuration

User Permissions

The USER_PERMISSIONS table stores currently licensed products and grants permissions to system features and functionality based on user types.

Licensed Products

Licensed products are listed with an action value of LICENSED.

System Mode

The USER_PERMISSIONS table can also define what modes a user type has access to. In this case, any of the following actions can be defined for a user type:

Value	Description
ACCESS_REAL_TIME	Allow the user to access the Real Time model.
ALL_EXECUTE, ALL_INSTRUCT	Some users can only execute switching steps inside certain types of blocks, such as Isolate. If so, they should be configured to Isolate_EXECUTE. Other users can execute steps in any block type (ALL_EXECUTE). Similar logic applies for instructing actions (ALL_INSTRUCT).
CREATE_STUDY_SESSIONS	Allow the user to create their own study sessions.

Value	Description
DMS_STUDY_MODE	Some users and user types do not automatically request PFService study sessions, which helps to conserve memory (in a service that is already memory intensive). DMS_STUDY_MODE provides a user or user type with automatic access to DMS in study mode. Other users will need to request it by checking the DMS Study checkbox.
USE_SHARED_STUDY_SESSION	Force the user to use a single shared study session called SHARED_STUDY_SESSION. Normally study sessions are named based on the user's login ID. Thus, this ID cannot be given to a user for login purposes. Also, since this option forces this user or user type to share a single study session, then this user or user type should have limited control to the Control Tool or any actions that result in a change to the model.

For Product configured Web Workspace:

- Start Real-time and Start Study mode buttons/menus are only visible when user has both CREATE_STUDY_SESSIONS and ACCESS_REAL_TIME permissions.
- Reset Study Session menu is only visible when user has CREATE_STUDY_SESSIONS permission.
- Close Study Session menu is only visible when user has both CREATE_STUDY_SESSIONS and ACCESS_REAL_TIME permissions.

Example:

```
INSERT INTO user_permissions (seq_permission_id, user_name, action)

VALUES( SEQ_USER_PERMISSION.NEXTVAL, 'Switching Entry',
'ACCESS_REAL_TIME');
```

The *Switching Entry* users cannot create study sessions and will be in a sense always stuck in Real Time. If the user type is given access to a viewer and control tool, projects should consider using the USE_SHARED_STUDY_SESSION option instead.

These options are in place to minimize the demand on the services to keep track of individual study sessions used by users. The more study sessions that are active, the higher the burden on the services. Keeping the study sessions to a minimum is highly recommended and these rules should be used to do that.

Web Workspace Editing Permissions

Web Workspace defines access rights to elements within a switching sheet based on user type. This allows certain parts of the application or switching sheet to be edited by one user type and not by other user types. For user type permissions within Web Workspace, the user_name should indicate the user type and the action should be the actions that user type has permissions over within the sheet.

Example:

```
INSERT INTO user_permissions (seq_permission_id, user_name, action)
VALUES( SEQ_USER_PERMISSION.NEXTVAL, 'Full Operations plus Web
Switching', 'ALL');
```

This would give the *Full Operations plus Web Switching* user type permissions over every aspect of a switching sheet. Use the **ALL** action to define this.

Example:

```
INSERT INTO user_permissions (seq_permission_id, user_name, action)
VALUES( SEQ_USER_PERMISSION.NEXTVAL, 'Switching Prep', 'SWITCHING');
INSERT INTO user_permissions (seq_permission_id, user_name, action)
VALUES( SEQ_USER_PERMISSION.NEXTVAL, 'Switching Prep',
'Isolate_EXECUTE');
```

This gives the *Switching Prep* user type permission to use Web Switching as well as execute permissions for steps, but only when the steps are in an *Isolate* block. The format of the action string is: `<Step Block Name>_<Step Action Name>`.

User Type Configuration

There are two approaches to configuring user environments by user types. The first method creates project configuration directories for each specific user type/role/privilege; the second sets rules within configuration files to enable different tools or views based on the user type/role/privilege. The first method works best if there are significant differences between the two versions; the second method is useful with small changes to the existing configuration.

In either method, user types must be defined in the **ENV_CODE** table.

ENV_CODE Table

Column	Data Type	Null-able	Comments
PRODUCT	VARCHAR2 (32 CHAR)	No	The current codes are: • `CREW` (Web Workspace) • `MODEL` (Model Management) • `OMS_CONFIG_TOOL` (Configuration Assistant) • `SERVICE_ALERT` (Service Alert) • `STORM` (Storm Management) • `WCB` (Web Callbacks) • `WCE` (Web Call Entry)
CODE_NAME	VARCHAR2 (32 CHAR)	No	Same as the environment above
CODE_SCRIPT	VARCHAR2 (32 CHAR)	Yes	The directory name of the overrides for this role
DISPATCH_GROUP	VARCHAR2 (32 CHAR)	Yes	Dispatch_groups.name
DGROUP_AUTH	VARCHAR2 (1 CHAR)	Yes	'Y'=allowed to change filtering dgroup

Example: NMS Standard ENV_CODE Table

Product	Code_Name	Code_Script
CREW	Administration	admin
CREW	Crew Operations	crew_ops
CREW	Full Operations	
CREW	Full Operations plus Switching	
CREW	Switching Request	request
CREW	Trainer	trainer
CREW	Trouble Maintenance	trbl_mnt
CREW	View Only	viewonly
MODEL	Model Validation	
OMS_CONFIG_TOOL	Administration	
SERVICE_ALERT	Administration	
STORM	Full Operations	
STORM	Storm Administration	admin
STORM	View Only	viewonly
WCB	Full Operations	
WCE	Web Call Entry	

Configuring User Roles with Subdirectories

1. Create a subdirectory of the project configuration directory that would be used for a specific user type/role/privilege (for example, view-only).
2. Copy each `*.xml` (or properties) file, which needs to be different for that user type, into this directory. Everything would be at the same level in the subdirectory; in other words, the directory would contain all `*.xml`, `*.inc`, and `*.properties` files that are specific for that user.
3. Edit the files to make the desired changes.
4. Have an entry in the `ENV_CODE` table for the user type and include the subdirectory name containing the configuration for the user type.

   For example, for the Crew Operations environment, the `ENV_CODE` table would use the following SQL statement:

   ```
   INSERT INTO env_code ( product, code_name, code_script,
   dispatch_group, dgroup_auth )
      VALUES ( 'CREW', 'Crew Operations', 'crew_ops', '', '' );
   ```

   The configuration files for the environment would be located in the `$NMS_BASE/[project]/jconfig/crew_ops` directory.

Example

Creating a view-only subdirectory and changing the viewer background color for them.

1. Create the view-only directory in the project configuration directory. For example: `$NMS_BASE/[project]/jconfig/ops/view-only/`.
2. Copy `VIEWER_GLOBAL_PROPERTIES.inc` from: `${NMS_BASE}/dist/baseconfig/product/ops/viewer/xml/` to `$NMS_BASE/[project]/jconfig/ops/view-only/`.
3. Modify the viewer.background_color line from:

   ```
   <StringProperty name="viewer.background_color"
    value="241,243,248"/>
   ```

   to

   ```
   <StringProperty name="viewer.background_color"
    value="0,0,0"/>
   ```

4. Run `nms-install-config --java`
5. Restart WebLogic.
6. Start Web Workspace, open a Viewer, and verify that the background color is now black.

User Role Constraints on Configuration

As in the other method, each application or tool that requires different access or a separate view for a user type will need to be configured for that user type.

1. If a project version of the tool configuration does not exist, copy it the appropriate project configuration directory.
2. You'll need to add restrictions, as appropriate, in the configuration files to turn on or off features for a user type.

Example: Restricting the ability to create a switching sheet for view only user.

```
<Menu name="MNU_FILE">
 <SubMenu name="MNU_NEW">
   <MenuItem name="MNU_NEW_SHEET" icon="new.png"
    accelerator="control N" hide_icon="true">

     <Visible initial="false" when="!USER_VIEW_ONLY and
      DS_LOGIN_ENTRY.WEB_SWITCHING_ENABLED == 'true'"/>

     <Enabled initial="false" when="!USER_VIEW_ONLY"/>

     <PressPerform>
       <Command value="DisplayNewNMSDialogCommand"
        when="DS_LOGIN_ENTRY.ENV == 'WEB' and
        DS_LOGIN_ENTRY.TYPE == '&UserSwitchingEntry;'">
          <Config name="dialog" value="DLG_NEW_NMS_DIALOG"/>
          <Config name="check_authority" value="false"/>
       </Command>

       <Command value="DisplayNewNMSDialogCommand"
        when="DS_LOGIN_ENTRY.ENV == 'WEB' and
        DS_LOGIN_ENTRY.TYPE != '&UserSwitchingEntry;'">
          <Config name="dialog" value="DLG_NEW_NMS_DIALOG"/>
       </Command>
     </PressPerform>

</MenuItem>
...
</Menu>
```

Login Tool Configuration

The **Login Tool** is responsible for determining which user type the user should log in as and verifying the password (if LDAP integration is turned off).

To configure an application (for example, Web Workspace or Configuration Assistant) to use the **Login Tool**, the `product_name` global property should be set in the tools configuration to the value as it exists in product column of the `ENV_CODE` table.

The following example demonstrates configuring Web Call Entry (**WCE**) to use the login tool:

```
<JBotTool width="830" height="900">
   <GlobalProperties>
      <StringProperty name="product_name" value="WCE"/>
   </GlobalProperties>
...
```

User Session Configuration

The **Login Tool** has a configuration option that handles user sessions when a user logs into the system from a different client.

Login Bean Properties

- **File**: `./jconfig/server/CentricityServer.properties`

```
LoginEJB.force_relogin = <true | false>
```

When set to false, if a user is currently logged into the application (or has abnormally exited within two minutes), the user will receive an error message saying the user is already logged in. The user can be released using the Configuration Assistant to reset the login.

When set to true, if another log in occurs for the same application and user, the original session will be automatically logged off. The system will not allow the user to save their work. The system will return a dialog box informing that the existing user was logged off, and that they should retry the log in. Clicking the **Login** button again will then log the user into the system and the new user (session) will begin.

Master Window Configuration (Oracle Fusion Client Platform)

The main client application window for Web Workspace can be configured by using a separate XML file. For example:

```
<dockingPositions xmlns="http://nms.oracle.com"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://nms.oracle.com http://
localhost/xml/docking.xsd"
             bounds="30,0,1220,942" maximized="false">
   <WEST floatSize="1003" dockSize="1210">
      <box>
         <leafBox height="300" width="400">
            <dockable dockWidth="400" floatHeight="300"
floatOrientation="-1" dockHeight="300"
ID="com.splwg.oms.client.crew.CrewIcons" floatWidth="400"/>
            <dockable dockWidth="400" floatHeight="300"
floatOrientation="-1" dockHeight="300"
ID="com.splwg.oms.client.authority.Authority" floatWidth="400"/>
         </leafBox>
      </box>
   </WEST>
   <EAST floatSize="180" dockSize="275">
      <box>
         <leafBox height="300" width="400">
            <dockable dockWidth="400" floatHeight="300"
floatOrientation="-1" dockHeight="300"
ID="com.splwg.oms.client.workspace.Workspace" floatWidth="400"/>
         </leafBox>
      </box>
   </EAST>
   <NORTH floatSize="180" dockSize="490">
      <box>
         <leafBox height="300" width="400">
            <dockable dockWidth="400" floatHeight="300"
floatOrientation="-1" dockHeight="300"
ID="com.splwg.oms.client.workagenda.WorkAgenda" floatWidth="400"/>
         </leafBox>
      </box>
   </NORTH>
   <SOUTH floatSize="180" dockSize="250"/>
</dockingPositions>
```

The file should be saved as the name of the application, with docking. For example, the name of the file for Web Workspace would be `Workspace_docking.xml`.

The easiest way to modify this file is to arrange the windows the way you wish, then exit the system. Bring up Windows Explorer, and locate `c:\Documents and Settings\[user]\.nms\system11.0.0.0.0\o.ide.11.1.1.1.33.53.67\windowinglayout.xml`

From that file, cut and paste the dockingPositions element to the configuration file. Note that the <dockingPositions> element should include the attributes listed above, although they are not in the `windowingLayout.xml`.

The "bounds" dockingPositions attribute specifies the position and size of the master frame. The numbers are the x, y, width, and height of the master frame.

The "maximized" attribute specifies if the frame should start maximized.

The default docking for applications that do not have a specific docking configuration is defined by jconfig/global/xml/JbotTool_docking.xml.

Java Web Start Page

Configure a $NMS_HOME/[project]/jconfig/license.properties file. Follow the instructions in the file to configure which projects to display on the login web page, as well as their titles.

Table Export Configuration

The Table Export feature allows you to set up an action to export table data to a CSV file. The table export behavior is defined in the TableBehavior element; if not defined, the export behavior will be inherited from the system default settings found in the CentricityTool.properties file.

TableBehavior Elements

- **copy_mode:** the copy_mode determines the behavior when the table is copied to the clipboard. Valid values:
  - TABLE: will copy the entire table, and CELL defines whether the entire table, the selected rows, or the selected cell is copied.
  - SELECTED: will only copy the selected rows
  - CELL: will only copy the selected cell.
- **copy_include_headers:** defines whether the header cells should be copied. Valid values: true or false.
- **copy_include_hidden:** defines whether hidden rows should be copied. Valid values: true or false.

Export to CSV

The Export Table functionality is assigned to a menu or button using the ExportToCSVCommand JBot command. In the following example, the Work Agenda's export command [3] exports the entire table [5], including headers [6], but not hidden rows [7] and sets the default save as name.

```
[1]  <MenuItem name="MNU_EXPORT_TABLE">
[2]    <PressPerform>
[3]      <Command value="ExportToCSVCommand">
[4]        <Config name="table" value="TBL_WA_ALARMS"/>
[5]        <Config name="mode" value="TABLE"/>
[6]        <Config name="include_headers" value="true"/>
[7]        <Config name="include_hidden" value="false"/>
[8]        <Config name="default_name" value="Workagenda.csv"/>
[9]      </Command>
[10]    </PressPerform>
[11] </MenuItem>
```

Work Agenda Configuration

Work Agenda configuration files are found in `${NMS_BASE}/dist/baseconfig/product/ops/workagenda/`.

GlobalProperties

The WORKAGENDA_GLOBAL_PROPERTIES.inc file contains the following tool properties:

String Properties

work_agenda.status_icon.##REPLACE.STATUS_<status>: Icon to display for the configured status. This can be configured for every status that needs an icon. Leave the value blank if no icon is desired.

Example:

```
<StringProperty name="work_agenda.status_icon.##REPLACE.STATUS_ASN"
value="qual_wrench_arrowup_16.svg"/>
<StringProperty name="work_agenda.status_icon.##REPLACE.STATUS_NEW"
value=""/>
```

Integer Properties

work_agenda.default_ert_offset: - If configured, the number of ms to use as the default ERT, as calculated from the current time

work_agenda.maximum_views: The maximum number of Work Agendas

work_agenda.reading_pane_population_delay: Delay in milliseconds between row selection in the Work Agenda treetable and start of the population of the Reading Pane panel for the selected row. Default: 1000 ms -->

work_agenda.requested_completed_outage_visible_time: The complete retain period requested from the server, for display of completed rows, in number of days

Boolean Properties

work_agenda.always_show_fuzzy_outages: Whether to display outages in the fuzzy zone, even if the user is not subscribed to the Fuzzy zone. Default: false

work_agenda.keep_restored_cmi: Whether to continue calculating the CMI for restored events. Default: false

work_agenda.show_completed_outages: Whether the tool should automatically load completed events. Default: false.

work_agenda.use_active_outage_as_lead: By default, a "No" answer in the WORKAGENDA.INCLUDE_SUB_EVENTS dialog will execute the action on the lead event, even if it's a non-outage. Set this to true to perform the action on the lead active outage event, when the lead event is a non-outage or is inactive. This also affects actions that don't display the dialog, like crew assignments.

String Array Properties

work_agenda.completed_testates: States that should be considered as completed

work_agenda.grouping_dialog_columns: List of DS_WA_ALARMS columns to be copied when event is added to the DS_EVENTS_TO_GROUP datastore. These columns will be available for display in the DLG_EVENT_GROUP (Group Events) dialog.

work_agenda.in_progress_testates: States to be considered as in progress

work_agenda.incomplete_testates: States to be considered as incomplete

work_agenda.summaryrow_feederclass: Device class named to be considered feeder heads, for the summary table

work_agenda.summaryrow_recloserclass: Device class named to be considered reclosers, for the summary table

work_agenda.summaryrow_fuseclass: Device class named to be considered fuses, for the summary table

work_agenda.to_do_testates: States to be considered as to do

Changing a Column Heading

Scenario: You want to change a column heading from Feeder to Circuit.

Column headings are defined in `${NMS_BASE}/dist/baseconfig/product/ops/workagenda/properties/WorkAgenda_en_US.properties` in the alarms panel section. You find the line:

```
################
# alarms panel #
################
...

TBL_WA_ALARMS.FEEDER_ALIAS.text = Feeder
```

1. Create a new blank **WorkAgenda_en_US.properties** file in the appropriate project or testing configuration directory (for example, `C:\OracleNMS\[your_project_name]\jconfig\ops\workagenda\properties\WorkAgenda_en_US.properties`).
2. Add the line: `TBL_WA_ALARMS.FEEDER_ALIAS.text = Circuit`
3. In a terminal window, enter the following:

   ```
   cd c:\OracleNMS\<your_project_name>\jconfig
   ```

4. Build the configuration using ANT:

   ```
   ant config
   ```

5. Log into Web Workspace to verify the column heading change. See Testing the Java Client Configuration on page 18-6 for details on configuring and starting Web Workspace in a testing environment.

Adding a Column

Scenario: You want to add a column to the Work Agenda.

If you wish to add a project specific column, do the following:

1. Add the column to the `JOBS` table.
2. Also add the column with the same name, datatype, and size to the `UNTIED_OUTAGES` table.
3. Add the column name to the `GENERIC_EVENT_FIELDS` table.
4. Restart services.

For further information, see Chapter 19, JMService Configuration on page 19-1.

The next step is to configure it to show up in Work Agenda:

1. Copy `${NMS_BASE}/dist/baseconfig/product/ops/workagenda/xml/WORKAGENDA_TBL_WA_ALARMS.inc` to `C:\OracleNMS\<your_project_name>\jconfig\ops\workagenda\xml\`.
2. Add a new line following the line:

   ```
   <Column key="CTRL_ZONE_NAME_4"/>.
   ```

3. On the new line, add:

```
<Column key="mds_id"/>
```

4. Open the project version of **WorkAgenda_en_US.properties**, created in the previous example.
5. Add the following lines:

```
TBL_WA_ALARMS.mds_id.text = MDS Id
TBL_WA_ALARMS.mds_id.width = 50
```

6. Build the configuration using ANT:

```
ant config
```

7. Log into Web Workspace to verify the column has been added.

Crew Actions Configuration

The Crew Actions Tool configuration files are found in `${NMS_BASE}/dist/baseconfig/product/ops/crew/`.

Crew Actions provides a mechanism for filtering crews based on control zone level; this filtering is capable of listening to the Work Agenda and filtering crews based on the event zone. The functionality is found in a Filter sub-menu description in `CREW_ICONS_MENUBAR.inc`:

```
<SubMenu name="MNU_FILTER_EVENT_ZONE">
    <RadioButtonMenuItem name="RBG_EVENT_ZONE_ALL" hide_icon="true"
     button_group="RBG_EVENT_ZONE"
     true_value="ALL" data_source="DS_EVENT_ZONE.LEVEL">
        <Enabled initial="true" when="ZONE_FILTER_ENABLED"/>
        <PressPerform>
            <Command value="FilterSelectedZoneCommand"/>
        </PressPerform>
    </RadioButtonMenuItem>
    <RadioButtonMenuItem name="RBG_EVENT_ZONE_DISTRICT"
hide_icon="true"
        button_group="RBG_EVENT_ZONE" true_value="CTRL_ZONE_NAME_2"
        data_source="DS_EVENT_ZONE.LEVEL">
        <Enabled initial="true" when="ZONE_FILTER_ENABLED"/>
        <PressPerform>
            <Command value="FilterSelectedZoneCommand"/>
        </PressPerform>
    </RadioButtonMenuItem>
   <RadioButtonMenuItem name="RBG_EVENT_ZONE_OFFICE" hide_icon="true"
     button_group="RBG_EVENT_ZONE"
     true_value="CTRL_ZONE_NAME_3" data_source="DS_EVENT_ZONE.LEVEL">
        <Enabled initial="true" when="ZONE_FILTER_ENABLED"/>
        <PressPerform>
            <Command value="FilterSelectedZoneCommand"/>
        </PressPerform>
    </RadioButtonMenuItem>
    <Enabled initial="true" when="ZONE_FILTER_ENABLED"/>
</SubMenu>
```

Related files and settings:

- The Work Agenda `WORKAGENDA_TBL_WA_ALARMS.inc` includes the `FilterSelectedZoneCommand` that interacts with Crew Actions to apply the filter in Crew Actions when one (or more) Work Agenda row(s) is selected.

- The Crew Actions `CREWICONS_TOOLBEHAVIOR.inc` also includes the `FilterSelectedZoneCommand`.
- `CREW_DATASTORES.inc` contains the datastore class name for the event zone: `<DataStoreClass name="DS_EVENT_ZONE"/>`
- The `CrewIcons.xml` configuration file sets the tool behavior when filtered in the `PNL_CrewFilters` sub-panel.

Removing the Event Filter Sub-menu

Scenario: Your project does not require filtering by event zone.

1. Copy `${NMS_BASE}/dist/baseconfig/product/ops/crew/xml/CREW_ICONS_MENUBAR.inc` to `C:\OracleNMS\[your_project_name]\jconfig\ops\crew\xml\`.
2. Remove the entire `<SubMenu name="MNU_FILTER_EVENT_ZONE">` section.
3. Run `ant config`.
4. Log into Web Workspace. The sub-menu will no longer be included under the Filter menu.

Event Details Configuration

Event Details configuration files are found in `${NMS_BASE}/dist/baseconfig/product/ops/eventdetails/`.

Adding a Drop-Down List

Scenario: You need to add a new drop-down list to the Event Details window. You will create a value called *animals* that will contain choices of *lions*, *tigers*, and *bears*.

Note that while the Configuration Assistant can add new values to an event details drop-down menu, it cannot create the new list. Therefore, for the first option, you need to add it directly to the database using SQL Developer or an alternative SQL tool of your choice.

1. Add the following entry to the `PICKLIST_GUI` table.

   ```
   Lion, animals_om, pushbutton, non_outage, 100
   ```

2. Add a column called `animals_om (VARCHAR2(20))`, to the `PICKLIST_INFO_UPD_TR` table.
3. Start Configuration Assistant. Select `animals_om` and add entries for Outage for lions, tigers, and bears. Select non-outage and add tigers and bears.
4. Copy the following files from:

   `${NMS_BASE}/dist/baseconfig/product/ops/eventdetails/xml/`

   to

   `C:\OracleNMS\<your_project_name>\jconfig\ops\eventdetails\xml\`:

   - `EVENTDETAILS_DATASTORES.inc`
   - `EVENTDETAILS_GLOBAL_PROPERTIES.inc`
   - `EVENTDETAILS_PNL_ACTIONS.inc`

5. To the copied file, **EVENTDETAILS_DATASTORES.inc**, add the following datastore:

```
<DataStoreClass name="DS_ANIMALS_OM"
class="com.splwg.oms.client.eventdetails.PicklistGUIDataStore"
table="picklist_gui"/>
```

6. Next, in the `EVENTDETAILS_GLOBAL_PROPERTIES.inc` file, find the line that defines `eventdetails.populate_datastores` and add `DS_ANIMALS_OM` to the list.
7. Next, find the `eventdetails.categories` section and add `ANIMALS_OM` to the list.
8. In the `EVENTDETAILS_PNL_ACTIONS.inc` file, add the following section before the `LBL_WEATHER` definition:

   Note: the ComboBox `display_menu` and submenu attributes should only be added when there are submenus defined using the Configuration Assistant.

```
<Label name="LBL_ANIMALS">
  <LabelPlacement start="2,0" height="1" width="1" weight="0,0"
   fill="NONE" insets="2,2,10,2" anchor="EAST"/>
</Label>
<ComboBox name="CMB_ANIMALS">
  <ComboBoxPlacement start="3,0" height="1" width="1"
   weight="1,0" fill="HORIZONTAL"/>
  <ComboBoxBehavior data_source="DS_EVENT_DETAILS.ANIMALS_OM"
   keys_data_source="DS_ANIMALS_OM.PANE_NAME"
  default_value="PROPERTY.UNSELECTED" display_menu="DISPLAY_MENU"
   submenu="SUBMENU">
    <Editable initial="false"/>
    <Enabled initial="false" when="!USER_VIEW_ONLY"/>
    <SelectPerform>
      <Command value="SetStatusFlagCommand">
        <Config name="flag_names" value="EVENT_DETAILS_EDITED" />
        <Config name="flag_values" value="true" />
      </Command>
    </SelectPerform>
  </ComboBoxBehavior>
</ComboBox>
```

9. Create, or append to, the file `EventDetails_en_US.properties` with the following:

```
LBL_ANIMALS.text = Animals
```

10. If a menu title is changed or added, the post completion log properties need to be updated. In `$NMS_CONFIG/jconfig/global/properties/JBotFormat_en_US.properties`, search for this section:

```
# Substitutions for field name in the Post-Completion Edit Log
FIELD_NAME.REF_ID = Event #
FIELD_NAME.EMERG_SW_ORDER_TEXT = Switching Plan #
FIELD_NAME.DEVICE_TEXT = Device
FIELD_NAME.CONTROL_ZONE_TEXT = Zone
FIELD_NAME.SYSTEM_OM = System
FIELD_NAME.CAUSE_OM = Sub-System
FIELD_NAME.TYPE_OM = Type
FIELD_NAME.FAILURE_OM = Failure
FIELD_NAME.INTERRUPT_DEV_OM = Interrupting Device
FIELD_NAME.ADV_WEATHER_OM = Weather
FIELD_NAME.ADV_ENVIRON_OM = Environment
FIELD_NAME.VEGETATION_OM = Vegetation
```

```
FIELD_NAME.FOREIGN_INTERF_OM = Foreign Interference
FIELD_NAME.DEF_EQUIP_OM = Defective Equipment
```

Update the names to reflect the names you are using.

11. Run `ant config`.
12. Log into Web Workspace. You should now see the new option when you launch the Event Details window for an event.

Adding a Validation Rule

Scenario: You want to create a validation rule that requires the user to choose an animal in order to close Event Details.

1. Copy **EVENTDETAILS_VALIDATION.inc** from: `${NMS_BASE}/dist/baseconfig/product/ops/eventdetails/xml/` to the project `eventdetails/xml` folder.
2. Add the following line to the copied file:

```
<ValueCheck group_names="VERIFY_COMPLETE_FORM" check_type="value"
fail_type="fail" ignore_blank="false" match_on="false"
values="Unselected" prompt="Must fill in a value for Animals"
widget_name="CMB_ANIMALS"/>
```

Note that in the **EVENTDETAILS_MENUBAR.inc** file, there is the following section which runs the validation:

```
<Command value="RunValidationCommand" when="!SWITCHING_EVENT">
  <Config name="group" value="VERIFY_COMPLETE_FORM"/>
</Command>
```

Update Events Configuration

The Update Events window is a Work Agenda dialog box that allows users to update Event Details information for multiple jobs at the same time. The Update Events configuration files are found in `${NMS_BASE}/dist/baseconfig/product/ops/workagenda/`.

Adding Drop-Down Lists

1. Copy **DLG_EVENTDETAILS_UPD.xml** from: `${NMS_BASE}/dist/baseconfig/product/ops/workagenda/xml/` to the project `workagenda/xml` folder.
2. Add local DataStores definitions to the copied DLG_EVENTDETAILS_UPD.xml for each of the OM datastores used in Event Details. These DataStores should be renamed with a _DLG suffix so they do not conflict with those used in Event Details.

For example:

```
<DataStoreClass name="DS_SYSTEM_OM_DLG"
class="com.splwg.oms.client.eventdetails.PicklistGUIDataStore"
table="picklist_gui" scope="local">
  <PrimaryKey value="PARENT_PANE"/>
  <PrimaryKey value="PICK_ENV"/>
</DataStoreClass>
```

3. Use these datastore names along with the categories in the calls to PopulateEventDetailsDlgCommand used in the dialog. These will usually correspond to the datastores and categories found in found in EVENTDETAILS_GLOBAL_PROPERTIES.inc.

For example:

```
<Command value="PopulateEventDetailsDlgCommand">
  <Config name="datastores"
  value="DS_SYSTEM_OM_DLG,DS_CAUSE_OM_DLG,DS_FAILURE_OM_DLG,DS_INTE
  RRUPT_DEV_OM_DLG,DS_TYPE_OM_DLG,DS_ADV_WEATHER_OM_DLG,DS_ADV_ENVI
  RON_OM_DLG,DS_VEGETATION_OM_DLG,DS_FOREIGN_INTERF_OM_DLG,DS_DEF_E
  QUIP_OM_DLG,DS_SCHEDULED_DEV_OM_DLG,DS_HUMAN_ELEM_OM_DLG,DS_OTHER
  _CAUSE_OM_DLG,DS_REMEDY_OM_DLG,DS_PRIMARY_CAUSE_OM_DLG"/>
  <Config name="categories"
  value="SYSTEM_OM,CAUSE_OM,FAILURE_OM,INTERRUPT_DEV_OM,TYPE_OM,ADV
  _WEATHER_OM,ADV_ENVIRON_OM,VEGETATION_OM,FOREIGN_INTERF_OM,DEF_EQ
  UIP_OM,SCHEDULED_DEV_OM,HUMAN_ELEM_OM,OTHER_CAUSE_OM,REMEDY_OM,PR
  IMARY_CAUSE_OM"/>
</Command>
```

4. Configure the drop-downs to point to the datastore, and also add an associated check box, which should be set to toggle a <category name>_CHECKED field in the miscellaneous datastore.

   For example:

```
<ComboBox name="CMB_SYSTEM">
  <ComboBoxPlacement start="1,0" height="1" width="1" weight="1,0"
  fill="HORIZONTAL"/>
  <ComboBoxBehavior data_source="DS_EVENT_DETAILS_DLG.SYSTEM_OM"
  keys_data_source="DS_SYSTEM_OM_DLG.PANE_NAME"
  default_value="PROPERTY.UNSELECTED">
    <Editable initial="false"/>
    <Enabled initial="true"
    when="DS_EVENT_DETAILS_MSCL_DLG.SYSTEM_OM_CHECKED == 'Y'"/>
    <SelectPerform>
      <Command value="SetStatusFlagCommand">
        <Config name="flag_names" value="EVENTDETAILS_UPD_MODIFIED"
        />
        <Config name="flag_values" value="true" />
      </Command>
    </SelectPerform>
  </ComboBoxBehavior>
</ComboBox>

  <CheckBox name="CHK_SYSTEM_OM">
    <CheckBoxPlacement start="2,0" fill="NONE" insets="2,9,2,10"
    weight="0,0" anchor="WEST"/>
    <CheckBoxBehavior initially_selected="true"
    data_source="DS_EVENT_DETAILS_MSCL_DLG.SYSTEM_OM_CHECKED">
    <Enabled initial="true"/>
    <PressPerform>
      <!-- Revert the value when we uncheck -->
      <Command value="SetDataStoreValuesCommand"
      when="DS_EVENT_DETAILS_MSCL_DLG.SYSTEM_OM_CHECKED == 'N'">
      <Config name="sources"
      value="DS_EVENT_DETAILS_DLG.system_om_original"/>
      <Config name="targets"
      value="DS_EVENT_DETAILS_DLG.SYSTEM_OM"/>
      </Command>
    </PressPerform>
  </CheckBoxBehavior>
</CheckBox>
```

5. Optionally, you can also update Equipment Failure fields by using the DS_EQUIP_FAIL_DLG. The Check box should update the <field>_CHECKED flag.

   For example:

```
<TextField name="TXTF_SERIAL">
  <TextFieldPlacement start="7,5" height="1" width="1" weight="1,0"
  fill="HORIZONTAL" insets="2,2,2,30" anchor="WEST"/>
  <TextFieldBehavior data_source="DS_EQUIP_FAIL_DLG.SERIAL_TEXT"
  modify_flag="EVENT_DETAILS_EDITED">
    <ValidValues type="string" max_characters="24"/>
    <Enabled initial="false"
    when="DS_EVENT_DETAILS_MSCL_DLG.SERIAL_TEXT_CHECKED == 'Y'"/>
  </TextFieldBehavior>
</TextField>
<CheckBox name="CHK_SERIAL_TEXT">
  <CheckBoxPlacement start="8,5" fill="NONE" insets="2,9,2,10"
  weight="0,0" anchor="WEST"/>
  <CheckBoxBehavior initially_selected="true"
  data_source="DS_EVENT_DETAILS_MSCL_DLG.SERIAL_TEXT_CHECKED">
    ...
  </CheckBoxBehavior>
</Checkbox>
```

6. If desired, you can also update generic JOBS table fields.

   For example:

```
<TextField name="TXTF_POWER_UP_COUNT">
  <TextFieldPlacement start="7,7" height="1" width="1" weight="1,0"
  fill="HORIZONTAL" insets="2,2,2,30" anchor="WEST"/>
  <TextFieldBehavior
  data_source="DS_EVENT_DETAILS_MSCL_DLG.power_up_count"
  modify_flag="EVENT_DETAILS_EDITED">
    <ValidValues type="string" max_characters="24"/>
    <Enabled initial="false"
    when="DS_EVENT_DETAILS_MSCL_DLG.power_up_count_CHECKED == 'Y'"/
    >
  </TextFieldBehavior>
</TextField>
<CheckBox name="CHK_POWER_UP_COUNT">
  <CheckBoxPlacement start="8,7" fill="NONE" insets="2,9,2,10"
  weight="0,0" anchor="WEST"/>
  <CheckBoxBehavior initially_selected="true"
  data_source="DS_EVENT_DETAILS_MSCL_DLG.power_up_count_CHECKED">
    <Enabled initial="true"/>
  </CheckBoxBehavior>
</CheckBox>
```

7. You can also configure other PICKLIST_INFO_UPD_TR fields.

   For example:

```
<CheckBox name="CB_NO_DTR_FLAG">
  <CheckBoxPlacement start="7,6" height="1" width="1" weight="0,0"
  fill="NONE" insets="2,0,2,2" anchor="WEST"/>
  <CheckBoxBehavior data_source="DS_EVENT_DETAILS_DLG.NO_DTR_FLAG"
  initially_selected="false">
    <Enabled initial="false"
    when="DS_EVENT_DETAILS_MSCL_DLG.NO_DTR_FLAG_CHECKED == 'Y'"/>
  </CheckBoxBehavior>
</CheckBox>
<CheckBox name="CHK_NO_DTR_FLAG">
```

```
  <CheckBoxPlacement start="8,6" fill="NONE" insets="2,9,2,10"
  weight="0,0" anchor="WEST"/>
  <CheckBoxBehavior initially_selected="true"
  data_source="DS_EVENT_DETAILS_MSCL_DLG.NO_DTR_FLAG_CHECKED">
  ...
  </CheckBoxBehavior>
</Checkbox>
```

Trouble Summary Configuration

The data that is displayed in Trouble Summary is populated by TSService on a timer controlled by the **-period** command-line option. Product configuration counts all outage, non-outage, and planned outage jobs. Change your version of the TJ_JOBS view to exclude other job types, if desired. The Trouble Summary offers the following configurable parameters.

refresh_period

Period of time (in seconds) between automatic updates of the Outage Summary information. This only makes the tool reload information from the database. It has no effect on how often TSService recalculates the data.

If set to 0 then periodic updating of Outage Summary is disabled.

Default: `600 seconds`

damage_population_delay

Delay in milliseconds between row selection in the Outage Summary tree-table and start of the population of the Damage Summary panel for the selected control zone.

Default: `1500 ms`

control_zone_separator

Character string used as a separator between individual control zone names when constructing full zone name. For example, OPAL/Stark/Lake would be the full zone name for the Lake control zone when `/` used as a separator.

Default: `'/'`

control_zone_depth

Number of control zone hierarchy levels to be displayed in Outage Summary table.

Value '-1' would cause full control zone tree to be displayed.

Default: 4

Viewer Configuration

Viewer configuration consists of defining the various model layers and defining application properties that control the Viewer behavior. The Viewer configuration is read by the application server and, consequently, updates to Viewer configuration require restarting WebLogic to deploy.

Adding a Separate Layer for SCADA Fuses

Viewer configuration consists of defining the various device layers and defining various properties that control the Viewer behavior. The layer configuration is read by the application server.

The layer definitions are found in `${NMS_BASE}/dist/baseconfig/product/ops/viewer/xml/SPATIALLAYERS_LAYERS.inc`.

Scenario: You want to remove all SCADA-controlled fuses from the Underground Fuses layer and add them to a new SCADA Fuses layer for display in the Viewer. (This scenario is based on the characteristics of the OPAL model.)

1. Copy **SPATIALLAYERS_LAYERS.inc** from: `${NMS_BASE}/dist/baseconfig/product/ops/viewer/xml/` to the project viewer/xml folder.
2. Search for the Underground Fuses layer definition:

```
<Layer name="Underground Fuses"
       active_on_start="true"
       screen_selectable="true"
       annotation_only="false"
       dxf_layer="true"
       condition_layer="false"
       electrical_layer="true"
       draw_order="6">
  <Class name="rack_fuse_ug_hd"/>
  <Class name="scada_fuse_ug_hd"/>
  <Class name="scada_rack_fuse_ug_hd"/>
  <Class name="fuse_ug_hd"/>
  <Class name="rack_fusr_ss_hd"/>
  <Class name="scada_fusr_ss_hd"/>
  <Class name="scada_rack_fusr_ss_hd"/>
  <Class name="fusr_ss_hd"/>
</Layer>
```

3. Copy the definition and paste the copy above the current definition. Edit the file to add a new SCADA Fuses layer definition and a modified Underground Fuses layer that does not include the SCADA fuse classes.

```
<Layer name="SCADA Fuses"
       active_on_start="true"
       screen_selectable="true"
       annotation_only="false"
       dxf_layer="true"
       condition_layer="false"
       electrical_layer="true"
       draw_order="6">
  <Class name="scada_fuse_ug_hd"/>
  <Class name="scada_rack_fuse_ug_hd"/>
  <Class name="scada_fusr_ss_hd"/>
  <Class name="scada_rack_fusr_ss_hd"/>
</Layer>
<Layer name="Underground Fuses"
       active_on_start="true"
       screen_selectable="true"
```

```
        annotation_only="false"
        dxf_layer="true"
        condition_layer="false"
        electrical_layer="true"
        draw_order="6">
    <Class name="rack_fuse_ug_hd"/>
    <Class name="fuse_ug_hd"/>
    <Class name="rack_fusr_ss_hd"/>
    <Class name="fusr_ss_hd"/>
  </Layer>
```

4. Copy **DLG_VIEWER_HIDE_DISPLAY_LAYERS.inc** from: `${NMS_BASE}/dist/baseconfig/product/ops/viewer/xml/` to the project `viewer/xml` folder. Edit the file to add the new hide/display settings:

   - Add a check box to toggle this new layer.

     ```
     <CheckBox name="CHK_SCADA_FUSES">
       <CheckBoxPlacement start="0,relative" weight="0,0"
        insets="0,0,0,0"/>
       <CheckBoxBehavior>
         <PressPerform>
           <Command value="ToggleLayersCommand">
             <Config name="layers" value="SCADA Fuses"/>
           </Command>
           <Command value="RedrawCommand"/>
         </PressPerform>
       </CheckBoxBehavior>
     </CheckBox>
     ```

   - Label the button:

     ```
     CHK_SCADA_FUSES.text = SCADA Fuses
     ```

5. Run **nms-install-config --java**
6. Restart WebLogic.
7. Start Web Workspace, open a Viewer, and start the Hide/Display tool to verify that SCADA fuses is listed as a layer.

Configuring Conductor Priority

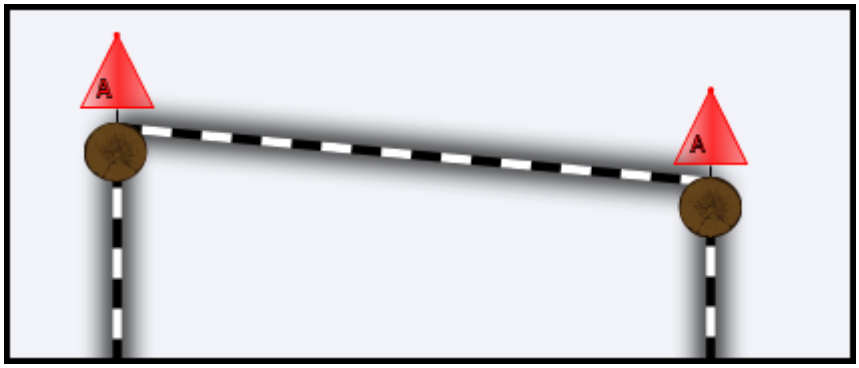
Conductor priority and highlight coloring is defined by `SPATIALLAYERS_CONDUCTOR_PRIORITY.inc`:

```
<ConductorPriority>
  <Code state="MULTISTATE"/>
  <Code state="FAULTED"/>
  <Code state="GROUNDED"/>
  <Code state="DELEGATED"/>
  <Code state="SECURE"/>
  <Code state="ISOLATED"/>
  <Code state="BIDIRECTIONAL"/>
  <Code state="PARALLEL"/>
  <Code state="MESH"/>
  <Code state="DEGRADED" highlight_color="BLACK"/>
  <Code state="PHASE"/>
  <Code state="MULTISTATE"/>
  <Code state="SUSPECT_OPEN"/>
  <!-- highlight deenergized areas with a BLACK highlight -->
  <Code state="DEENERGIZED" highlight_color="BLACK"/>
  <Code state="ENERGIZED"/>
  <Code state="DIST_GEN" highlight_color="LIMEGREEN"/>
```

```
</ConductorPriority>
```

The codes are listed in priority order. Therefore, if a conductor is both `FAULTED` (third line) and `GROUNDED` (fourth line), the conductor will color as `FAULTED`.

The `highlight_color` is an optional definition of the highlight that will appear around the conductor with the indicated status. The highlight color will display for the first matching status with a highlight color, regardless of the main conductor color being used. If no status with a `highlight_color` matches, no highlighting will occur. All RGB colors are supported.



Configuring Big Symbols for Digital Measurements Symbols

If you wish to set big symbols for digital measurement symbols, add the following `Symbol` definition to SPATIAL_LAYERS_BIG_SYMBOLS.xml:

```
<Symbol class="digital" scale=".25"/>
```

If you wish to only set big symbols for some measurement keys or if you wish to use different scales for different symbols, define each symbol using the following syntax:

```
<Symbol class="digital:#" scale=".25"/>
```

where `#` is the measurement key from the SCADA system. For example:

```
<Symbol class="digital:2" scale=".15"/>
<Symbol class="digital:3" scale=".25"/>
```

> **Note:** you cannot set a separate definition for a particular symbol by adding it separately from the `<Symbol class="digital" scale=".25"/>` definition. For example, the following is **not** allowed:
>
> ```
> <Symbol class="digital" scale=".25"/>
> <Symbol class="digital:3" scale=".15"/>
> ```

Changing the Viewer Background Color

The Viewer's GUI configuration is defined in `${NMS_BASE}/dist/baseconfig/product/ops/viewer/xml/`**VIEWER_GLOBAL_PROPERTIES.inc**.

Scenario: You want to change the background color of the Viewer drawing area.

1. Copy **VIEWER_GLOBAL_PROPERTIES.inc** from: `${NMS_BASE}/dist/baseconfig/product/ops/viewer/xml/ to the project viewer/xml folder.`
2. Modify the viewer.background_color line from:

   ```
   <StringProperty name="viewer.background_color"
    value="241,243,248"/>
   ```

   to

   ```
   <StringProperty name="viewer.background_color"
    value="black"/>
   ```

3. Run **nms-install-config --java**
4. Restart WebLogic.
5. Start Web Workspace, open a Viewer, and verify that the background color is now black.

Configuring Hide-Display Conditions

The Hide/Display dialog can be configured to have the Viewer show (or not show) conditions based upon attributes or condition status.

> **Note:** For technical reasons, crew status values must subtract one. So to match a crew status 3, you would use "2" as the value.

1. In VIEWER_HIDE_DISPLAY_CONDITIONS_SETUP.inc, add lines defining the status that should be controlled.

   For example, adding *Service* and *Trouble* crew types to one layer:

   ```
   <Command value="SetupConditionFilterCommand">
       <Config name="column" value="crew_0_1"/>
       <Config name="class" value="truck_location"/>
       <Config name="status" value="0,1"/>
   </Command>
   <Command value="SetupConditionFilterCommand">
       <Config name="column" value="crew_2"/>
       <Config name="class" value="truck_location"/>
       <Config name="status" value="2"/>
   </Command>
   <Command value="SetupConditionFilterCommand">
       <Config name="column" value="crew_3"/>
       <Config name="class" value="truck_location"/>
       <Config name="status" value="3"/>
   </Command>
   <Command value="SetupConditionFilterCommand">
       <Config name="column" value="crew_4"/>
       <Config name="class" value="truck_location"/>
       <Config name="status" value="4"/>
   </Command>
   ```

   where

   - *column* is an arbitrary name in the DS_VIEWER_DEFAULT datastore that the checkbox will be bound to.
   - *class* is the class of the condition.

- *status* is a comma delimited list of statuses.

2. Modify DLG_VIEWER_HIDE_DISPLAY_CONDITIONS.inc to match this configuration:

```
<SubPanel name="PNL_CREWS_SUB_PANEL">
  <PanelPlacement start="0,250" weight="1,0" insets="2,2,2,2"/>
  <CheckBox name="CHBOX_HD_CREW_0_1" >
    <CheckBoxPlacement start="0,relative" weight="0,0"
insets="0,0,0,0" />
    <CheckBoxBehavior data_source="DS_VIEWER_DEFAULT.crew_0_1"
data_source_value_type="true/false" initially_selected="true">
      <PressPerform>
        <Command value="RefreshCommand"/>
      </PressPerform>
    </CheckBoxBehavior>
  </CheckBox>
   <CheckBox name="CHBOX_HD_CREW_1" >
    <CheckBoxPlacement start="0,relative" weight="0,0"
insets="0,0,0,0" />
    <CheckBoxBehavior data_source="DS_VIEWER_DEFAULT.crew_1"
data_source_value_type="true/false" initially_selected="true">
      <PressPerform>
        <Command value="RefreshCommand"/>
      </PressPerform>
    </CheckBoxBehavior>
  </CheckBox>
  <CheckBox name="CHBOX_HD_CREW_2" >
    <CheckBoxPlacement start="0,relative" weight="0,0"
insets="0,0,0,0" />
    <CheckBoxBehavior data_source="DS_VIEWER_DEFAULT.crew_2"
data_source_value_type="true/false" initially_selected="true">
      <PressPerform>
        <Command value="RefreshCommand"/>
      </PressPerform>
    </CheckBoxBehavior>
  </CheckBox>
  <CheckBox name="CHBOX_HD_CREW_3" >
    <CheckBoxPlacement start="0,relative" weight="0,0"
insets="0,0,0,0" />
    <CheckBoxBehavior data_source="DS_VIEWER_DEFAULT.crew_3"
data_source_value_type="true/false" initially_selected="true">
      <PressPerform>
        <Command value="RefreshCommand"/>
      </PressPerform>
    </CheckBoxBehavior>
  </CheckBox>
  <CheckBox name="CHBOX_HD_CREW_4" >
    <CheckBoxPlacement start="0,relative" weight="0,0"
insets="0,0,0,0" />
    <CheckBoxBehavior data_source="DS_VIEWER_DEFAULT.crew_4"
data_source_value_type="true/false" initially_selected="true">
      <PressPerform>
        <Command value="RefreshCommand"/>
      </PressPerform>
    </CheckBoxBehavior>
  </CheckBox>
</SubPanel>
```

The key is to configure the data_source, which should match the previous step. The **RefreshCommand** will refresh the viewer with the new settings.

Configure Search Options

Adding a New Search

Add the search entry datastore. The name of the datastore must be in the format of DS_VIEWER_XX_SEARCH_DATA.

In VIEWER_SEARCH_DATASTORES.inc, add the line:

```
<DataStoreClass name="DS_VIEWER_MY_SEARCH_DATA" scope="local"/>
```

Next add the text field(s) for the user to enter the search criteria to DLG_VIEWER_SEARCH (modify a copy of an existing search):

```
<SubPanel name="PNL_SEARCH_BY_MY">
  <PanelPlacement start="2,relative" height="1" width="2" weight="1,1"
  insets="2,2,5,2" fill="HORIZONTAL" anchor="CENTER"/>
  <Label name="LBL_LEFT_DUMMY_SPACER">
  <LabelPlacement start="0,0" width="1" height="1" weight="0,1"
  fill="BOTH"/>
  </Label>
  <Label name="LBL_RIGHT_SPACER">
  <LabelPlacement start="4,0" width="1" height="1" weight="0,1"
  fill="BOTH" ipad="3,0"/>
  </Label>
  <Label name="LBL_SEARCH_TEXT_MY>
  <LabelPlacement start="1,0" insets="2,2,0,2" anchor="WEST"
  fill="HORIZONTAL"/>
  </Label>

  <TextField name="TXTF_VIEWER_SEARCH_MY_NAME">
    <TextFieldPlacement start="3,relative" height="1" weight="0,0"
    insets="1,1,1,1" fill="HORIZONTAL" anchor="WEST"/>
    <TextFieldBehavior columns="20"
    data_source="DS_VIEWER_MY_SEARCH_DATA.MY_NAME ">
    <ReturnPerform>
    <Command value="ExecuteActionCommand">
       <Config name="action" value="SEARCH_ACTION"/>
    </Command>
    </ReturnPerform>
    <Perform name="Focus" category="focusGained">
    <Command value="ExecuteActionCommand">
    <Config name="action" value="CLEAR_SEARCHES"/>
    <Config name="$ENABLED_FLAG$" value="MY_PARAM_ENABLED"/>
    </Command>
    </Perform>
    <ValidValues max_characters="32"/>
    </TextFieldBehavior>
  </TextField>
</SubPanel>
```

Then add this to the SEARCH_ACTION section:

```
<Command value="ViewerSearchCommand" when="MY_PARAM_ENABLED">
  <Config name="dialog" value="DLG_VIEWER_MY_SEARCH_RESULT"/>
  <Config name="query" value=
    "select h_cls, h_idx, my_name from my_table
       where
    $(my_name: DS_VIEWER_MY_SEARCH_DATA.MY_NAME)
    order by my_name"/>
  <Config name="device_alias" value="my_name"/>
  <Config name="device_handle" value="h_cls.h_idx"/>
  <Config name="partition_handle" value="p_cls.p_idx"/>
</Command>
```

The config options are:

- **query:** The query to perform. Text in $(…) indicates replacements.
- **device_handle:** Columns of the handle of the device. If the device class and index are in columns h_cls, and h_idx, the value will be: "h_cls.h_idx"
- **device_alias:** The column of the alias of the device. The search can work with either the handle or the alias of the device. If the device_alias is configured, it will use it for the displayed alias in the viewer.
- **partition_handle:** The columns of the device partition. It is in the format of p_cls.p_idx. If the device_handle is included but not the partition hande, it will focus on the first partition.
- **coordinates:** If two coordinates are provided, it represents the upper left corner and lower right corners of the bounding box. otherwise it indicates the center.
- **coord_system:** The column containing coordinate system to search. This is an optional parameter only used when the coordinates are specified. If not defined it will use 0.
- **area_name:** The column containing the name of the area or intersection that should be displayed as the name in the viewer.

More information about the **ViewerSearchCommand** is available in the online JBot Command Reference; see page 18-14 for information on the online documentation.

Next, determine if you can use an existing _SEARCH_RESULT dialog or you need a new one. You can use an existing dialog if you the result set is compatible with an existing dialog. For example, if you have a special query that returns certain devices, you can use DLG_VIEWER_DEVICE_SEARCH_RESULT. However, if you wish to display different columns in the result, you will need to create a custom result dialog. In that case, copy and modify a dialog that is close to what you require.

Ensure that the dialog you require is configured in the ViewerSearchCommand under the "dialog" parameter.

Replacement Rules

The above matching syntax (`$(my_name: DS_VIEWER_MY_SEARCH_DATA.MY_NAME`) will translate into a **like**, **equals**, **soundex**, or **upper()** version of the columns as needed. However, if there is a need to not follow the sort options, but search for something directly, just use the datastore name as a replacement value without the colon:

```
<Config name="query" value=
   "select h_cls, h_idx, my_name from my_table
   where
     my_name = $(DS_VIEWER_MY_SEARCH_DATA.MY_NAME)
   order by my_name"/>
```

Configuring Condition Tooltips

When the mouse is hovered over a condition symbol (that is, a tag, note, document, call, clue, or damage assessment), it will display information about the condition. This can be modified by using a custom Java class that extends com.splwg.oms.client.viewer.Tooltip. See the example, `demo.CustomTooltip`, as part of the OPAL project. Variables are specified by enclosing them within ${}. They can either refer to text that is included in Viewer_en_US.properties, or else the datastore associated with the condition. To see all the available columns in the datastore run a datastore report (Creating the JBot DataStore Report on page 18-30).

Configuring a Specific Trace Type

If you desire additional Trace To <device> options, add them to the DLG_VIEWER_TRACE.xml in the following way:

1. Add a new keyword to the CMB_TRACE_TO ComboBox.
2. In the ACT_TRACE definition, add a <Command> that executes when DS_VIEWER_DEFAULT.TRACE_TO matches your new keyword. Pass the desired device class name as the "device_cls" argument to this Command.
3. Label the option in the DLG_VIEWER_TRACE_en_US.properties file using CMB_TRACE_TO.<keyword>.text

For example, we will use the new keyword **EXAMPLE** and a device class called **example_device_cls**.

In DLG_VIEWER_TRACE.xml

```
<ComboBox name="CMB_TRACE_TO">
...
+      <Key value="EXAMPLE"/>
...
<Action name="ACT_TRACE">
...
+   <Command value="TraceCommand" when="DS_VIEWER_DEFAULT.TRACE_TO ==
'EXAMPLE'">
+       <Config name="type" value="TRACE_TO_DEVICE_CLASS"/>
+       <Config name="device_cls" value="example_device_cls"/>
+   </Command>
...
```

In DLG_VIEWER_TRACE_en_US.properties:

```
+ CMB_TRACE_TO.EXAMPLE.text = Example Device Class
```

Configuring Viewer Map Loading Limits

There are a few limits that work together:

```
<IntegerProperty name="viewer.max_maps_error" value="550"/>
```

This will be the largest number of maps that the client will try to request from the server.

```
<IntegerProperty name="viewer.max_devices" value="100000"/>
```

This is the largest number of devices that can be returned by the server. The server keeps a tally of the number of objects as it puts together the map list, and if it is exceeded, it returns an error. This means that there is some extra load put on the server to load the maps (especially if the maps have not been loaded before), but, because it fails before trying to send the results back to the client, it is still rather quick.

Finally, there is the WebLogic **maxmessagesize**, which is the maximum size of an object that can be transferred. This results in a failure after returning 50 MG to the client (or whatever the max is set to by the project).

The point of these settings is to keep the user from choosing too many maps, and minimizing the impact if too many are chosen. If the first limit is hit, there is no load on the server. If the second is hit, there is some load, but the server gives up before things get unreasonable. You do want to avoid routinely hitting the **maxmessagesize**; you will receive a more generic error message, and the server goes through all the work to return the maps, only to fail when the return size is too large.

You can determine how large the maps are by various debug settings.

Enabling the **warn_if_over_bytes** setting in CentricityTool.properties will log the size of large messages requests. You can see if the return values are close to the 50 MB default **maxmessagesize**.

Another helpful setting is the VIEWER_TIMING client debug, which Bug 26307332 added printing the number of object and vertices downloaded. This can be enabled through the help menu.

One thing to note: You really do need to experiment with those limits to determine the appropriate settings for max maps.

If a project has maps with less objects and vertices per map, then the limits can be safely raised. If the maps are large and many vertices, then the product values might be too high. So you cannot assume that the product values are safe (and raising them is risk) so you need to find a limit that works for the project.

Configuring Viewer Tooltips

Tooltips are rendered using html of various datastore values. The configuration can be modified by overriding Tooltip.java. To use a project specific configuration modify this line in the viewer configuration:

```
<StringProperty name="viewer.tooltip_class"
value="com.splwg.oms.client.viewer.Tooltip"/>
```

There is an example in the opal configuration, which is a copy of the product configuration in `OPAL/jconfig/java/src/demo/CustomTootip.java`.

So to use that version, copy CustomTooltip.java to your project and modify the viewer configuration to:

```
<StringProperty name="viewer.tooltip_class"
value="demo.CustomTooltip"/>
```

See CustomTooltip.java for details.

Trending Graphs Configuration

The data to display, most labels, and date ranges, etc, are configurable using standard JBot commands. However, if a project wants to customize the graph further, it can be done by overriding the Oracle JET components by saving project specific versions to $NMS_CONFIG/jconfig/public_html.

The files that generally could be overridden are:

- js/views/chart.html
- js/views/outage_chart.html
- js/viewModels/chart.js
- js/viewModels/outage_chart.js

The JBot Commands that populate this are PopulateGraphCommand and PopulateOutageGraphCommand.

Feeder Load Management Configuration

Feeder Load Management Global Properties may be modified as follows:

- **max_flm_tools:** Set the maximum number of tools that may be started; default value is 2.
- **refresh_minutes:** Set the number of minutes before the calculations are refreshed; default value is 3 using the IntegerProperty. Fractional values are allowed using the DoubleProperty element; see example that follows.

Scenario: Configure Feeder Load Management to refresh every 90 seconds.

- The product configuration sets refresh_minutes using the IntegerProperty element. The refresh time for this scenario is equal to 1.5 minutes, a fractional time that is not valid for an IntegerProperty.

1. Copy **FLMSummary.xml** from: `${NMS_BASE}/dist/baseconfig/product/ops/flm/xml/` to the project `flm/xml` folder.
2. Open the new product version of FLMSummary.xml.
3. In the Global properties element:

```
<GlobalProperties>
   <StringProperty name="product_name" value="FLM"/>
   <IntegerProperty name="max_flm_tools" value="2"/>
   <!-- How often you allow the FLM Summary to automatically
        refresh, in minutes. -->
   <IntegerProperty name="refresh_minutes" value="3"/>
</GlobalProperties>
```

4. Change :

```
<IntegerProperty name="refresh_minutes" value="3"/>
```

to

```
<DoubleProperty name="refresh_minutes" value="1.5"/>
```

Model Management Application Configuration

The Model Management application provides a user interface for viewing status of model build related events (full builds, patches, and pending maps) and initiating a model build scripts. The tool may be configured to run the model build scripts in the background or to run them synchronously and display the result in a dialog.

Scenario: You want to configure the Model Management application to rerun model build patches in the background.

- This scenario relies on the behavior of the BuildPatchCommand command. Whenever the command is called, you may pass the nohup command (to make the command run in the background) along with the script to run. Model Management has two files that utilize the BuildPatchCommand:
  - `DLG_BUILD_MAP.xml`: controls the actions when **Build Map...** is selected from the Model Management tool's **Action** menu.
  - `ModelManagement.xml`: the primary configuration file for the Model Management tool; includes the menu action for building a patch.

1. Copy ModelManagement.xml from: `${NMS_BASE}/dist/baseconfig/product/ops/model_management/xml/` to the project `model_management/xml` folder.
2. Add `nohup` to the MNU_RESUBMIT_PATCH popup menu command:

```
<PopupMenuItem name="MNU_RESUBMIT_PATCH"
   class="javax.swing.JMenuItem">
   <Enabled initial="false" when="TBL_PATCHES_SELECTED"/>
   <PressPerform>
      <Command value="BuildPatchCommand">
         <Config name="command" value="nohup nms-build-maps"/>
         <Config name="arg_01" value="-noVerify"/>
         <Config name="datastore" value="DS_PATCHES"/>
      </Command>
   </PressPerform>
</PopupMenuItem>
```

> **Note:** this example uses the standard `nms-build-maps` script, but you may substitute a custom version by copying the script to the project scripts folder and modifying that file.

DMS Summary Tool Configuration Guide

Rule Name	Valid values	Description
LEFT_PANE_DEVICES_QUERY	String	This is query to get devices from DB to display in left pane. The summary of these devices is shown in summary pane.
NO_OF_SUM_BUTTONS	Number Value	This is the number of different summaries to display in Summary panel
SUMMARY_QUERIES	Comma Delimited List of SQL query names	This is the list of SQL queries that need to be executed to get various summaries to display in summary panel
SUM_BTN.x.title	String	This is the title to display for the x button. Where, x is between 1 and NO_OF_SUM_BUTTONS. For example: SUM_BTN.1.title = Grid Load
SUM_BTN.x.num_desc	Number Value	This is the number of different descriptions to display in button x. Where, num_desc is between 1 and 3. For example: SUM_BTN.1.num_desc = 3
SUM_BTN.x.desc.y	String	This is the string for description y for button x. Where y is between 1 and num_desc. For example: SUM_BTN.1.desc.1=Energized: {#DER_ON} Where #DER_ON is a column defined in DS_SUMMARY data store. When the DMS Summary window is displayed and if the default format is followed, then the window shows the summary button SUM_BTN1 with a description something like: Energized: 6 / 8
SUM_BTN.x.DEV_QUERY	String	This is the SQL query to be executed to get the list of devices for the summary button x when it is selected.

Example of summary definition in DMS Summary properties file.

```
LEFT_PANE_DEVICES_QUERY=SUMMARY_FDRS_QUERY
NO_OF_SUM_BUTTONS=6
SUMMARY_QUERIES=SUMMARY_GRID_SOURCE_QUERY,SUMMARY_DER_QUERY,SUMMARY_FD
R_LOAD_QUERY,SUMMARY_SHUNT_CAPS_QUERY,SUMMARY_SWITCHES_QUERY,SUMMARY_X
FMRS_QUERY
SUM_BTN.1.title=Grid Load
SUM_BTN.1.num_desc=1
SUM_BTN.1.desc.1=Total Load: {SRC_OUTPUT} MVA
SUM_BTN.1.DEV_QUERY=SUMMARY_GRID_SOURCE_DEV_QUERY

SUM_BTN.2.title=DERs
SUM_BTN.2.num_desc=3
SUM_BTN.2.desc.1=Energized: {#DER_ON}
SUM_BTN.2.desc.2=Capacity: {DER_RATED_SIZE} kVA
SUM_BTN.2.desc.3=Output: {DER_OUTPUT} kVA
SUM_BTN.2.DEV_QUERY=SUMMARY_DER_DEV_QUERY
```

The result of summary queries are stored in DS_SUMMARY data store and the column names are output of the queries executed. All columns are stored as string values. You can either use them as it is or provide your own definition like #DER_ON as below definition.

```
<DataStoreClass name="DS_SUMMARY"
class="com.splwg.oms.jbot.DefaultDataStore" scope="local">
     <Column name="#DER_ON"
definition="#DER_ON,DER_CNT%JBotFormat.DMS_SUMMARY.X_OUT_OF_Y"/>
</DataStoreClass>
```

Understanding DmsSummaryParameters.properties File

All SQL queries that are being used by DMS Summary tool are defined in this file. You need to make sure these queries are executable without errors. The queries are having placeholders that are replaced by required data by the tool at run time as described below. Alternatively, you can replace the placeholders with the required data.

`$USER_CONTROL_ZONES$`

This is replaced by the control zones for which the user is subscribed to.

`$SELECTED_FEEDERS_IN_CLAUSE$`

This is replaced by the devices (feeders) selected in the left pane of the DMS Summary tool.

> **Note:** The tool will not work as expected if the names of these placeholders are changed.

Right-To-Left Language Configuration

Application configuration for Right-To-Left languages, such as Arabic, is configured through the CentricityTool.properties and the ImageLocalize.properties files.

Note that symbology file tooltips may be localized by editing the **.sym** files. See Chapter 8, Building the System Data Model, for details on sym files.

CentricityTool.properties

Text direction is defined in `${NMS_BASE}/dist/baseconfig/global/properties/CentricityTool.properties`.

1. Copy **CentricityTool.properties** from: `${NMS_BASE}/dist/baseconfig/product/global/properties/` to the project `global/properties` folder.
2. Modify the text.direction line from:

   ```
   text.direction = LTR
   ```

   to

   ```
   text.direction = RTL
   ```

ImageLocalize.properties

The ImageLocalize.properties file is used to override the XML files that define images/icons. The file is located in `${NMS_BASE}/dist/baseconfig/global/properties/`. The file has defined image files that will be flipped if RTL is set in CentricityTool.properties. You may also substitute other files by adding a substitution statement.

Scenario: You want to substitute an image for the info_ena.png ( ) file.

1. Copy **ImageLocalize.properties** to the project `global/properties` folder.
2. Edit the file:

   ```
   # The following lists those file that should be flipped or changed
   # when displaying in another language.("flip" is used for "RTL"
   # with arrows)

   info_ena.png = your_file.png

   oracle/javatools/icons/navigateBack.png = flip
   oracle/javatools/icons/navigateForward.png = flip
   textBigger.gif = flip
   textSmaller.gif = flip
   ```

Customizing Applications

Applications may be extended by providing custom commands, which may then be configured as part of the application. Additionally, NMS commands may be called from an external systems.

This section contains the following topics:

- Customization Examples using the Demo Tool
- Creating Custom Functions for Displaying Data
- Using Additional Libraries
- Invoking Commands from an External System
- Invoking Commands Using a Web Service

Customization Examples using the Demo Tool

Introduction

Prerequisites

This assumes the user is familiar with programming in Java and with Oracle Utilities Network Management System configuration.

The demo commands and tool are included as part of the OPAL configuration. Therefore, this documentation assumes that either the OPAL model is used or all the demo tools and configuration are copied to the correct project directory.

Setup

To run these examples, the following should be added to WorkspaceMenuBarTool. This will add a button to Web Workspace to display the demo tool:

```
<MenuItem name="MNUITM_DEMO" hide_icon="true">
  <PressPerform>
    <Command value="DisplayToolCommand">
      <Config name="tool" value="DemoTool"/>
      <Config name="class" value="com.splwg.oms.jbot.JBotTool"/>
    </Command>
  </PressPerform>
</MenuItem>
```

Using the Demo Tool

The Demo Tool provides examples for various text fields, a table, and buttons that demonstrate how to integrate a custom application into an Oracle Utilities Network Management System.

- The Hello World example displays a dialog.
- The AddCommand example adds two numbers and saves them in a third field.
- The IncrementCommand example shows how to access and change data.
- The DemoFocusCommand example shows how to call existing JBot commands.

The example code is in `$NMS_CONFIG/jconfig/java/src`. This is where any custom commands should be saved.

Access to data in Oracle Utilities Network Management System is saved in datastores, which are bound to the actual java swing components.

The demo tool is saved to `$NMS_CONFIG/jconfig/ops/test/xml/DemoTool.xml`.

Using the Demo Tool Sample Code

Hello World

The Hello World example code provides a simple command to display a dialog box displaying text:

See $NMS_CONFIG/jconfig/java/src/demo/HelloWorldCommand.java:

```
package demo;
import com.splwg.oms.jbot.JBotCommand;
import javax.swing.JOptionPane;
public class HelloWorldCommand extends JBotCommand {
    public void execute() {
        JOptionPane.showMessageDialog(null, "Hello World!");
    }
}
```

AddCommand

The AddCommand example provides a command that reads two values from the system and saves the sum to a third value.

```
package demo;

import com.splwg.oms.jbot.JBotCommand;

import java.awt.AWTEvent;
import java.awt.Component;

import javax.swing.JOptionPane;

/**
 * This command adds two numbers
 */
public class AddCommand extends JBotCommand {
    public void execute() {
        // This parameter must exist or else an error will occur
        String var1 = getRequiredParameter("var1");

        // If this parameter does not exist, the value will be null
        String var2 = getParameter("var2");

        String result = getRequiredParameter("result");

        double retVal;
        try {
            String number1 = (String)getDataSourceValue(var1);
            retVal = Double.parseDouble(number1);
            if (var2 != null) {
                String number2 = (String)getDataSourceValue(var2);
                retVal += Double.parseDouble(number2);
            }
            setDataSourceValue(result, retVal);
        } catch (Exception e) {
            AWTEvent awtEvent = (AWTEvent)getJBotEvent().getEvent();
            Component component = (Component)awtEvent.getSource();
            JOptionPane.showMessageDialog(component,
            "Could not add the numbers", "Error",
            JOptionPane.ERROR_MESSAGE);
            setAbort(true);
        }
    }
}
```

IncrementCommand

The IncrementCommand example shows how to read and write to multiple rows in a datastore:

package demo;

```
import com.splwg.oms.jbot.IDataRow;
import com.splwg.oms.jbot.IDataStore;
import com.splwg.oms.jbot.JBotCommand;

import java.awt.AWTEvent;
import java.awt.Component;

import javax.swing.JOptionPane;

/**
 * This example show how to access and update a datastore that
 * has multiple rows.
 */
public class IncrementCommand extends JBotCommand {
    public void execute() {
        IDataStore ds = getDataStore("DS_DEMO_TABLE");
        synchronized(ds.getLockObject()) {
            for (IDataRow row : ds) {
                Integer count = (Integer) row.getValue("count");
            row.setValue("count", Integer.valueOf(count + 1));
}
        }
        ds.notifyObservers();
    }
}
```

DemoFocusCommand

The DemoFocusCommand is an example on how to call existing JBot commands from within a custom JBot command. The example can be used to focus on a device in the viewer:

```
package demo;

import com.splwg.oms.client.viewer.FocusOnHandleCommand;
import com.splwg.oms.jbot.JBotCommand;
import com.splwg.oms.jbot.JBotException;

import java.util.HashMap;

/**
 * This is an example of calling an existing JBot command. It will
 * focus on a device with a given handle, given a datastore
 * values of the class and index of the device handle.
 */
public class DemoFocusCommand extends JBotCommand {
    public void execute() {
      String dataSource = getRequiredParameter("handle");
      String handleStr = getDataSourceValue(dataSource).toString();
      int pos = handleStr.indexOf(".");
      if (pos == 0) {
          throw new JBotException("Invalid handle");
      }
      String handleCls = handleStr.substring(0, pos);
      String handleIdx = handleStr.substring(pos+1);

      HashMap map = new HashMap();
```

```
        map.put("handle_cls", handleCls);
        map.put("handle_idx", handleIdx);

    // the options to this are the command name, a map of parameters
    // (or null if it doesn't take parameters, and the source for this
    // command (which can normally be left as null)
        getEnv().getTool().getAdapter()
         .runCommand(FocusOnHandleCommand.class.getName(), map, null);
    }
}
```

Creating Custom Functions for Displaying Data

There are times when you want to display data that integrates information from another system, or format data in a way that calculated fields do not have the flexibility to display. In this case, calculated functions can be used.

The following is an example of calling a custom function:

```
<Column name="#ReverseFeeder"
definition="#demo.ReverseFormat(FEEDER_ALIAS)%{0}"/>
```

Normally, items to the left of the % are the list of columns. Custom functions take those columns and perform an operation on them. For example, the above call reverses the characters in the first column listed.

Function ReverseFormat Source

```
package demo;
import com.splwg.oms.jbot.CustomFormat;

public class ReverseFormat extends CustomFormat {

    public Object[] format(Object[] source) {
        String input = (String)source[0];
        char[] chars = new char[input.length()];
        for (int i=0; i < chars.length; i++) {
            chars[chars.length - 1 - i] = input.charAt(i);
        }
        source[0] = new String(chars);
        return source;
    }
}
```

Custom Formats

NMS provides the following custom formats:

- **MatcherFormat**: Uses regular expressions to parse a string.
- **MatchSubstringFormat**: Returns a string if it contains a certian string, otherwise returns an empty value.
- **UserLookupFormat**: Returns the full username for a given userid.
- **MultiplyFormat**: allows you to add column multipliers to any DataStore that includes numbers.

See CustomFormat in the Javadocs for further information.

Using Additional Libraries

If additional client libraries are needed, they should be saved to $NMS_CONFIG/java/lib. Any jar files in this directory will be unjarred, and included as part of nms_config.jar.

Invoking Commands from an External System

Commands can be invoked by sending high level messages, either by using the "Action" command or by using a web service. (It is recommended that the web service be used for production use).

A listener for a high level message is defined as follows:

```
<Perform name="HLM" category="onMessage" type="DISPLAY_MESSAGE">
  <Command value="demo.DisplayMessageCommand"/>
</Perform>
```

This should be defined in the ToolBehavior portion of the tool you wish to integrate with.

The **type** is an arbitrary identifier of the action.

This can be invoked by running the following from the Oracle Utilities Network Management System server:

```
Action -java USER.* DISPLAY_MESSAGE "Hello world"
```

USER should be replaced with the username of the nms user.

This configuration calls the following command:

```
package demo;

import com.splwg.oms.jbot.HLMEvent;
import com.splwg.oms.jbot.JBotCommand;

import java.util.List;

import javax.swing.JOptionPane;

/**
 * This displays a message to the user from an external system
 */
public class DisplayMessageCommand extends JBotCommand {

    public void execute() {
        HLMEvent hlmEvent = (HLMEvent) getEvent();
        List<String> args = hlmEvent.getMessage().getArgs();
        String message = args.get(0);
        JOptionPane.showMessageDialog(null, message);
    }
}
```

Invoking Commands Using a Web Service

This should be invoked by using the sendHLM web service message.

The wsdl for the web service is located as follows:

- For WebLogic:

```
http://nms-server:7001/MessageBean/MessageBeanService?wsdl
```

(Replace nms-server with the dns name or IP address of the Oracle Utilities Network Management System system to connect to.)

Elements of the sendHLM Message

- **String_1:** username (used to determine destination environment)
- **HLMessage_2/from:** sender

  Note: the sender must be a valid user in the WebLogic security realm with the NmsService role.
- **HLMessage_2/message:** command
- **HLMessage_2/to:** destination tool
- **HLMessage_2/windowName:** not used
- **boolean_3:** synchronous message flag (if true, then web service call will return only after the message has been delivered).

The following command, which delivers the message 'DISPLAY_MESSAGE "Hello world"' to the tool 'demo' running in the environment of user 'user',

```
Action -java user.demo DISPLAY_MESSAGE "Hello world"
```

corresponds to the following web service message

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:mes="http://oms.splwg.com/ws/message">
  <soapenv:Header/>
  <soapenv:Body>
     <mes:sendHLM>
        <String_1>user</String_1>
        <HLMessage_2>
           <from>test</from>
           <message>DISPLAY_MESSAGE &quot;Hello world&quot;</message>
           <to>demo</to>
        </HLMessage_2>
        <boolean_3>true</boolean_3>
     </mes:sendHLM>
  </soapenv:Body>
</soapenv:Envelope>
```

Communicating with Other Programs Using Named Pipes

Clients can communicate with other applications running on the local box by using named pipes. This can be used to communicate with a SCADA application, for example. The pipes have to be set up by the other application. See the commands ReadPipeCommand, WritePipeCommand, and ActionFromHLMCommand for details.

Messages Available Using the OSI SCADA Adapter

If OSI SCADA is licensed (scada_osi has been added to `user_permissions` with a LICENSED action), the following actions are available as part of product.

Note: it is possible to use the following actions with other systems, if configured.

Messages from SCADA to NMS

- `MEASURE #DEVICE_ALIAS#=<ALIAS> #RTI_ALIAS#=<RTI ALIAS>`

  Displays the SCADA Summary for the given <ALIAS> and the given <RTI ALIAS>

- `SEARCH_BY_ALIAS #DEVICE_ALIAS=<ALIAS>`

  Displays <ALIAS> in viewer 1.

- `SEARCH_BY_ALIAS_FOR_VIEWER #DEVICE_ALIAS=<ALIAS> #VIEWER#=<VIEWER #>`

  Displays alias in the chosen viewer. <VIEWER#> should be 1 or 2.

Messages from NMS to SCADA

The SCADA Summary includes the following messages that a SCADA supplier may implement:

- Navigate to SCADA:

  ```
  ViewTrend point={DS_SCADA_SUMMARY.rtiAlias} minutes=60
  device=true
  ```

- Navigate to SCADA Alarm by Point:

  ```
  ViewAlarms point={DS_SCADA_SUMMARY.rtiAlias}
  ```

- Navigate to SCADA Alarm by Station:

  ```
  ViewAlarms station={DS_SCADA_SUMMARY.zoneName4}
  ```

- Navigate to SCADA Summary by Point:

  ```
  ViewTabular point={DS_SCADA_SUMMARY.rtiAlias} detail=false
  ```

- Navigate to SCADA Summary - Detail:

  ```
  ViewTabular point={DS_SCADA_SUMMARY.rtiAlias} detail=true
  ```

JBotCommand Methods Reference

The following commands may be called from a JBot command:

Method	Description
getParameter	protected java.lang.String **getParameter**(java.lang.String key) This returns the value of a configuration option for this command.
getDefaultmeter	protected java.lang.String **getDefaultParameter**(java.lang.String key, java.lang.String defaultValue)
getBooleanParameter	protected boolean **getBooleanParameter**(java.lang.String key, boolean defaultValue)
getRequiredBooleanParameter	protected boolean **getRequiredBooleanParameter**(java.lang.String key)
getRequiredParameter	protected java.lang.String **getRequiredParameter**(java.lang.String key) This returns the value of a configuration option for this command. If it does not exist, it throws a JBotException.
getParameterSubset	protected java.util.SortedMap<java.lang.String,java.lang.String> **getParameterSubset**(java.lang.String prefix) This will return the parameters in alphabetical order that start with the given prefix. **Parameters** prefix: Prefix of the parameter to match. **Returns** A sorted map of parameters
execute	public abstract void execute() throws java.lang.Exception This is the method invoked by the CommandProcessor when the command is executed. **Throws** java.lang.Exception
getName	public java.lang.String getName() Returns command String key. **Returns** java.lang.String
supressBusyCursor	public boolean supressBusyCursor() Return True if this command should not display the hourglass. This should only be set to true if the command is very fast.

Method	Description

`getEvent`	`public java.lang.Object getEvent()`
	
	Returns original event object. It could be any swing events for example.
	
	Returns
	`java.lang.Object`

`setStatusFlag`	`public void setStatusFlag(java.lang.String flag, boolean status)` Set the specified status flag in the DataManager. These statuses determine validation, JButtons' enabled status, etc. **Parameters** `flag`: the status value `status`: the boolean status
`getStatusFlag`	`public boolean getStatusFlag(java.lang.String flag)` Get the value of the specified status flag in the DataManager. These statuses determine validation, JButtons' status, an so forth. **Parameters** `flag`: the status value **Returns** • `True`: if flag is true • `False`: if flag not found or flag is false.
`fireStatusChanges`	`public void fireStatusChanges()` Notifies all interested components that the tool's statuses have changed.
`getDataStore`	`public IDataStore getDataStore(java.lang.String dataStoreKey)` Returns the DataStore with the specified key. **Parameters** `dataStoreKey`: the String key that describes the DataStore **Returns** The datastore.
`getCurrentDataRow`	`public final IDataRow getCurrentDataRow(java.lang.String dataStore)` A convenience method that will get the current datarow of a datastore. **Parameters** dataStore: the name of the data store.

Method	Description

`getJBotEvent`	`public JBotEvent getJBotEvent()`
	
	Returns JBotEvent object.
	
	Returns
	`com.ces.jbot.JBotEvent`

`isAbort`	`public boolean isAbort()` Indicates whether processing of additional commands in this package should be aborted.
`setAbort`	`protected void setAbort(boolean b)` If true, instructs the command processor to not process any additional commands for this event.
`getDataSourceValue`	`protected java.lang.Object getDataSourceValue(java.lang.String dataSource)` Returns the value of a datasource in the form of `[datastore].[column name]`. **Parameters** `dataSource`: the datasource **Returns** the value

Chapter 19

JMService Configuration

JMService Configuration includes the following topics:

- Trouble Code Configuration
- SRS Clues and Priorities
- Call Quality Configuration
- Generic Event Fields
- Weighted Customer Count
- Appointments
- Dispatch Groups Configuration

Trouble Code Configuration

SRS_TROUBLE_CODES Schema

Column Name	Column Type	Description
group_name	VARCHAR2(20)	name of trouble code group as displayed in Web Call Entry
group_order	INTEGER	group number to which trouble code entry belongs
code_name	VARCHAR2(40)	name of trouble code entry displayed in Web Call Entry
code_num	INTEGER	trouble code entry number (unique within each trouble code group)
priority	INTEGER	incident priority associated with this trouble code entry VALID VALUES: any non-negative integer
description	VARCHAR2(70)	description of the trouble code entry
short_desc	VARCHAR2(50)	short description of trouble code entry
groupable	INTEGER	0 if trouble code entry is not groupable, 1 if trouble code entry is groupable, 3 if trouble code entry is group into only. VALID VALUES: [0, 1, 3]
clue	INTEGER	1 if trouble code entry is a clue, 0 otherwise VALID VALUES: [0, 1]

Column Name	Column Type	Description
combine_pri	INTEGER	priority component used to compute combined priorities VALID VALUES: any non-negative integer
code_type	VARCHAR2(31)	code type. Normally 'default' unless trouble codes are hidden from Web Call Entry

Configuration

For normal operations, one or more trouble code groups must be configured. Each trouble code group consists of one or more trouble code entries. Each row in srs_trouble_codes corresponds to a single trouble code entry.

A trouble code is the grouping of one trouble code entry from each of the different trouble code groups.

Example

The following (incomplete) configuration has 2 trouble code groups. The first trouble code group is called "Lights" and has the entries "Unselected", "On", "Off", and "Blinking". The second trouble code group is called "Cause" and has the entries "Unselected", "Wire Down", and "Pole Broken".
Valid trouble codes with this configuration include *Lights="On" Cause="Unselected"*, *Lights="Off" Cause="Wire Down"*, and *Lights="Blinking" Cause="Unselected"*.

```
INSERT INTO srs_trouble_codes(group_name, group_order, code_name,
code_num)
VALUES('Lights', 1, 'Unselected', 0);
INSERT INTO srs_trouble_codes(group_name, group_order, code_name,
code_num)
VALUES('Lights', 1, 'On', 1);
INSERT INTO srs_trouble_codes(group_name, group_order, code_name,
code_num)
VALUES('Lights', 1, 'Off', 2);
INSERT INTO srs_trouble_codes(group_name, group_order, code_name,
code_num)
VALUES('Lights', 1, 'Blinking', 3);

INSERT INTO srs_trouble_codes(group_name, group_order, code_name,
code_num)
VALUES('Cause', 2, 'Unselected', 0);
INSERT INTO srs_trouble_codes(group_name, group_order, code_name,
code_num)
VALUES('Cause', 2, 'Wire Down', 1);
INSERT INTO srs_trouble_codes(group_name, group_order, code_name,
code_num)
VALUES('Cause', 2, 'Pole Broken', 2);
INSERT INTO srs_trouble_codes(group_name, group_order, code_name,
code_num)
```

Groups and Codes

Trouble code groups must be assigned consecutive integers starting with 1. All of the trouble code entries in a group must have the same group name. No two groups may have the same group name.

Trouble code entries within a group must be assigned consecutive code numbers starting with 0 or 1. Trouble code entries within a group must have distinct code names.

Other Fields

priority

Used in calculation of the priority of a call

description

This field is not used in any trouble code calculations

short_desc

The short description of a trouble code is equal to the distinct short_desc values for each of the selected trouble code entries

groupable

A trouble code's groupable value is 0 if any of its trouble code entries has a groupable value of 0. Otherwise, a trouble code's groupable value is equal to the highest groupable value among its trouble code entries. A groupable value of 0 means that the call is not groupable and will always create its own event. A groupable value of 1 means that the call can group to an existing groupable event or have other calls group to it. A groupable value of 3 means that the call will not group into other calls/events, but its event can have calls with groupable=1 grouped into it. 3 is primarily used for damage assessment.

clue

A trouble code is considered a clue if any of its trouble code entries has clue set to 1.

combine_pri

Used in calculation of the total priority of a call

SRS Clues and Priorities

What are Clues?

Clues are a subset of all trouble calls (incidents). Clues for a particular customer are whatever that customer has defined to be clues. There are two ways to define which trouble calls are clues:

- CLUE format item in a trouble call itself with a corresponding value of 1 will make the trouble call a clue.
- If any of the trouble codes of a call has been defined in the srs_trouble_codes table to be a clue, then the trouble call will be a clue.

The reason why clues are beneficial is that they have their own SRS broadcast rule **broadcastClues**. All trouble calls, clues and non-clues alike, will be broadcast by JMService to the outside world (viewer, interfaces, etc.) if the **broadcastIncidents** SRS rule is set. This can generate a lot of traffic and can clutter the viewer badly.

By defining as clues a subset of calls that add value to the system and that should be displayed, and by setting the **broadcastIncidents** SRS rule to 0 and **broadcastClues** to 1, only what the customer has defined as clues will be broadcast and thus displayed by the viewer.

The priority and condition status attributes of clues are also computed differently from those of normal trouble calls, as can be seen in the description below:

The Total Priority of a Trouble Call

The total priority of a trouble call is the sum of the priorities of the individual trouble codes of the trouble call. The total priority of a trouble call can be specified in three ways:

1. COMBINE_PRI format item in a trouble call itself with a corresponding value will use that value as the total priority.
2. If the COMBINE_PRI format item is not sent, the total priority will be computed by JMService as the sum of the priorities of the individual trouble codes of the trouble call.
3. A default total priority of 2 is used for "trouble calls" created by CreateEventCommand.

The Priority of a Trouble Call

The lower the value for priority, the higher the priority of the call. There are 4 ways to set the priority of a trouble call:

1. CUST_PRIORITY format item in a trouble call specifies the priority of the customer calling. If cusPriority SRS rule is used to specify the mappings from customer priorities to SRS priorities, then the value corresponding to this format item will be used to set the priority of the trouble call UNLESS rule 2 described below provides a smaller priority value (higher priority).
2. CUST_TROUBLE_CODE format item in a trouble call specifies the trouble codes of the call. priority column in srs_trouble_codes specifies the priority value of each trouble code. The lowest priority value of the trouble codes of the call will be used UNLESS rule 1 described above provides a smaller priority value (higher priority).
3. If the trouble call is a clue or if the useTotalPriority SRS rule is set, the priority of the trouble call is set to the total priority of the trouble call. This rule will override rules 1 - 2.
4. A default priority of 2 is used for "trouble calls" created by CreateEventCommand.

The Total Priority of an Event

The total priority of an event is the sum of the priorities of the unique trouble codes for all of the trouble calls that comprise an event.

Example:

The priority of the 'LTS OUT' trouble call has been previously determined to have a value of 3500. The priority of a 'LTS OUT-BROKEN POLE' trouble call has been previously determined to have a value of 6300.

By default, and event comprised of 5 'LTS OUT' and 1 'LTS OUT-BROKEN POLE' calls would have a total priority value of 9800 (3500 + 6300).

This default behavior can be overridden to calculate the total priority of the event to be the sum of ALL trouble call priorities by setting the sumOfAllTcodePri SRS rule. With that rule set, the example above would then be calculated to have a total priority value of 23800 (3500 x 5 + 6300).

The Condition Status of a Trouble Call

The condition status of a trouble call is used to determine what symbol to display for an outage. A symbol must be defined for every possible condition status. The condition status of a trouble call (incident) can be computed in the following four ways:

1. CUST_STATUS format item in a trouble call itself with a corresponding value will use that value as the condition status.
2. If the CUST_STATUS format item is not sent and the trouble call IS a clue, the condition status will be the highest combined priority value of the trouble codes of the trouble call. The combined priority values of trouble codes are specified in the combine_pri column of the srs_trouble_codes table.
3. If the CUST_STATUS format item is not sent and the trouble call is NOT a clue, the condition status will be the priority of the trouble call.
4. A default condition status of 2 is used for "trouble calls" created by CreateEventCommand.

What Calls Are Priority Calls?

Priority calls are calls that have a priority value (computed as detailed above) within the interval defined by **priorityCallMin** and **priorityCallMax** SRS rules.

Many customers display on their Work Agenda the total number of priority calls. Others may want to display their priority calls in different groups. This is achieved by specifying entries of trouble code matching criteria in the call_quality table for up to 4 different groups (pri_w, pri_sw, pri_p, pri_e).

Furthermore, some customers want an extra alarm row for each priority call on the Work Agenda. This is achieved by setting the **priorityIncAlarm** SRS rule.

Call Quality Configuration

Schema

CALL_QUALITY Schema

Column Name	Column Type	Description
Numb	INTEGER	Unique integer to distinguish rows in this table (PRIMARY KEY)
set_name	VARCHAR2(32)	rule set name (key)
tc_group_name	VARCHAR2(20)	trouble code group category (group_name in srs_trouble_codes)
tc_code_name	VARCHAR2(40)	trouble code name (code_name in srs_trouble_codes)
cust_priority	VARCHAR2(10)	customer priority from ces_customers
operation	VARCHAR2(10)	A value of "AND" allows two or more rows to be joined by the same op_group. A value of "ANY" does wildcard matching for trouble codes. A value of "AND,ANY" does both of these.
op_group	INTEGER	grouping number allowing two or more rows to be joined
rule_set	VARCHAR2(32)	The rule set when this rule applies. Leave this empty if the configuration should apply to all rule sets. If the behavior should differ between rule sets, add multiple rows, one for each rule set. Currently only supported for trouble_queue rows.

Set Types

Trouble Code Sets

These sets will match customer calls based on their trouble code (CUST_TROUBLE_CODE JMSinput field). If the CUST_TROUBLE_CODE matches the trouble code values (tc_group_name and tc_code_name) of a particular set or the trouble code combination (see below) of a particular set, the call is considered a member of that set. tc_group_name and tc_code_name must match the group_name and code_name of a row in the srs_trouble_codes table. If the operation column is set to "ANY", all other trouble code groups can be set to any value. Otherwise, all other trouble code groups must be set to their default values. If the operation column is set to "AND", a call quality row will be grouped with all other rows with operation set to "AND" and having the same set_name and group_order to create a trouble code combination. All specified trouble codes must match and all other trouble codes must be set to their default values for a trouble code combination to match a call's trouble code.

auto_upgrade - Calls matching this set will automatically cause a probable service outage to upgrade to a probable device outage.

cancel_call - Calls matching this set will cancel an existing call. This set takes priority over all other call sets.

dgroup_category - Calls matching this set will create events with their dispatch group category initially set to the value in the cust_priority field. See Dispatch Groups Configuration for more information.

exclude - Calls matching this set will be excluded from the call counts for their jobs.

make_damage - Calls matching this set will automatically create Hazard-type Damage Assessments on the device using the "damageTroubleCodeHazard" SRS Rule value for the trouble code. Values in this set usually match the pri_wire set, which is described below.

momentary - Calls matching this set will create probable momentary outages.

no_analysis - Calls matching this set will not create events/outages or be analyzed by the prediction engine. They will be saved to the database as completed. This set takes priority over all other sets except cancel_call.

non_electric - Calls matching this set will create non-electric events rather than outages.

non_outage - Calls matching this set will create non-outage events rather than outages.

pri_extreme - Calls matching this set will count towards the pri_e count of an event.

pri_pole - Calls matching this set will count towards the pri_p count of an event.

pri_service_wire - Calls matching this set will count towards the pri_sw count of an event.

pri_wire - Calls matching this set will count towards the pri_w count of an event.

priority_asn_alarm - This set is used in conjunction with the priorityAssignmentTimer and priorityAssignmentEnterState SRS rules. A call must match this set in addition to having these rules turned on for a Priority Assignment alarm to be issued.

so_grouping - This set is used in conjunction with the allowSOGrouping SRS rule. Calls must match this set in addition to having the rule turned on to group. If the cust_priority field is empty, then calls can only group if they have identical trouble codes. Otherwise, they can only group if both calls match the so_grouping set and either 1) they have the same cust_priority 2) one of them has a cust_priority of '*'.

still_out - If this set is configured, then when a Not Restored callback is processed, the trouble code of the original call will be masked/modified with the value in this set. If the operation is set for wildcard matching, then any unspecified trouble code groups will be retain their data from the original call's trouble code.

trouble_queue - Calls matching this set will create events with their work queue initially set to the value in the cust_priority field. Multiple records of this type can be configured for different rule_set values.

Priority Sets

These sets will match customer calls based on their priority value (CUST_PRIORITY JMSinput field). If the cust_priority column of one of these sets matches the CUST_PRIORITY for the call, the call is considered a member of that set. The tc_group_name, tc_code_name, operation, and op_group columns are not used for these sets.

crit_1 - Calls matching this set will count towards the crit_1 and crit_tot counts of an event.

crit_2 - Calls matching this set and not the crit_1 set will count towards the crit_2 and crit_tot counts of an event.

crit_3 - Calls matching this set and not the crit_1 or crit_2 sets will count towards the crit_3 and crit_tot counts of an event.

crit_tot - Calls matching this set will count towards the crit_tot count of an event.

Examples

```
/* Calls with a priority value of '2' will be counted as crit_1 and
crit_tot */
INSERT INTO call_quality(numb, set_name, tc_group_name, tc_code_name,
                         cust_priority, operation, op_group)
VALUES(1, 'crit_1', '', '', '2', '', 0);

/* Calls with a priority value of '1' will be counted as crit_2 and
crit_tot */
INSERT INTO call_quality(numb, set_name, tc_group_name, tc_code_name,
                         cust_priority, operation, op_group)
VALUES(2, 'crit_2', '', '', '1', '', 0);

/* Calls with trouble code group 'Power' set to 'Power On' will be
considered
   non-outage calls regardless of other trouble code selections */
INSERT INTO call_quality(numb, set_name, tc_group_name, tc_code_name,
                         cust_priority, operation, op_group)
VALUES(3, 'non_outage', 'Power', 'Power On', '', 'ANY', 0);

/* Calls with trouble code group 'Other' set to 'Cancel' and all other
trouble
   code options set to their defaults will be considered cancel calls
*/
INSERT INTO call_quality(numb, set_name, tc_group_name, tc_code_name,
                         cust_priority, operation, op_group)
VALUES(4, 'non_outage', 'Other', 'Cancel', '', '', 0);

/* Calls with trouble code group 'Power' set to 'Power Off', group
'Cause' set
   to 'Wire Down Pole to Pole' and all other trouble code options set
to their
   defaults will have their dispatch group category initially set to
   'X' */
INSERT INTO call_quality(numb, set_name, tc_group_name, tc_code_name,
                         cust_priority, operation, op_group)
VALUES(5, 'dgroup_category', 'Power', 'Power Off', 'X', 'AND', 1);
INSERT INTO call_quality(numb, set_name, tc_group_name, tc_code_name,
                         cust_priority, operation, op_group)
VALUES(6, 'dgroup_category', 'Cause', 'Wire Down Pole to Pole', 'X',
'AND', 1);
```

Generic Event Fields

A generic event field is arbitrary text information (limited in size to the size of corresponding column in JOBS table) associated with an event and identified by field name. Example configuration steps illustrated by configuring field 'generic1' which will allow to store strings up to 64 characters long.

Configure Generic Fields

All generic event fields to be used have to be listed in the GENERIC_EVENT_FIELDS database table. Field names are not case sensitive so having two fields which differs only by case is not allowed.

Structure of the GENERIC_EVENT_FIELDS database table:

Column Name	Description
FIELD_NAME	Generic field name. Used to identify field. Has to be unique.
FIELD_ORDER	Generic field ordinal number. Defines order in which fields are registered with WorkAgenda.
ATT_COLUMN	Name of the column in the attribute column. If not an empty string then generic field will be auto-populated with the information from the column in the attribute table for the outage device.

Example:
Add field 'generic1' to the GENERIC_EVENT_FIELDS table.

```
 INSERT INTO generic_event_fields (field_name, field_order,
att_column)
    VALUES ('generic1', 10, NULL);
```

Modify JOBS Table

A column of type VARCHAR2 matching name of the generic field configured in GENERIC_EVENT_FIELDS table should be added to the JOBS and UNTIED_OUTAGES tables if information in the generic field should be persistent. For example, fields populated from attribute tables might not need to be stored in the JOBS table. Size of the column in JOBS table will be used to check and truncate (if needed) values before updating database. The size of the corresponding UNTIED_OUTAGES column must match this size.

Example:
Add column 'generic1' to the JOBS table.

```
/* Add to the JOBS and UNTIED_OUTAGES table definitions
   generic1 VARCHAR2(64) */

ALTER TABLE jobs ADD generic1 VARCHAR2(64);
ALTER TABLE untied_outages ADD generic1 VARCHAR2(64);
```

Add Columns to Work Agenda

Add column to WORKAGENDA_TBL_WA_ALARMS.inc with key matching the name of the column and matching properties for column label and width in WorkAgenda_en_US.properties (See also Work Agenda Configuration on page 18-40.)

Example:
Add column 'generic1' to WORKAGENDA_TBL_WA_ALARMS.inc.

```
<Column key="generic1"/>
```

Add column 'generic1' to WorkAgenda_en_US.properties.

```
TBL_WA_ALARMS.generic1.text = Generic Field 1
TBL_WA_ALARMS.generic1.width = 60
```

Weighted Customer Count

Introduction

Purpose

A utility might have a need in major events to prioritize work and make it visible to the dispatchers. Some utility customers may have a higher priority in restoration; these include 911 centers, hospitals, police stations, schools, etc. During major events, outages with larger numbers of customers may obscure these priority customers from the dispatcher's view. With weighted customer count configured, the dispatcher could sort by the weighted customer count and dispatch the most important outages first.

Overview

The weighted customer counts feature is designed to address the problem of prioritizing work. Each customer has "weight" value associated with its account. The weighted customer count of an outage in the system is the sum of the "weights" of all customers affected by the outage. The weighted customer count for an outage is available in Work Agenda and can be used for sorting.

The customer weight value for an individual account can come from one of two sources:

1. Base Customer Weight Value
   All customers are divided into groups based upon the value of a single column in the customer model. This column will be referred to as the base weight key. The customer weight value is determined by the group a specific customer belongs to.
2. Customer Weight Override
   The customer weight value entered by administrator (via Configuration Assistant) for a specific account.

The customer weight override takes precedence, if available.

Configuration

Customer Schema

Project-specific additions to the customer model are required to support weighted customer counts.

1. Column WEIGHTED_COUNT must be added to the CES_CUSTOMERS view as an INTEGER. This provides the base weight of the customer.
2. Column WEIGHTED_COUNT must be added to the CUSTOMER_SUM view. It should be populated by adding together the base customer weight values of all customers on a supply node.

SRS Rules

The rule useWeightedCounts must be enabled.

Work Agenda Configuration

The column WEIGHTED_NUM_CUST can be added to Work Agenda (See **Adding a Column** in **Work Agenda Configuration**) and/or configured in pre-defined sorts.

Appointments

Appointment Types

The appointment types that appear in Web Call Entry and Work Agenda are configured in the OMS_APPOINTMENT_TYPES table.

OMS_APPOINTMENT_TYPES Schema

Column Name	Type	Description
APPT_TYPE	INTEGER NOT NULL	Appointment type. Primary key
DISPLAY_ORDER	INTEGER	Order in which appointments types are displayed
SHORT_DESC	VARCHAR2(32) NOT NULL	Short description of appointment type. Used for display to user.
DESCRIPTION	VARCHAR2(256)	Long description of appointment type.

Dispatch Groups Configuration

Dispatch groups are a way to categorize jobs. These categories (called referral groups) can be automatically assigned based on the type of call that creates the job, job's device, and the job's control zone. Users can change a job's referral group with UpdateReferralCommand.

Tables

DISPATCH_GROUP_CATEGORIES

This table contains the set of valid dispatch group categories. Jobs created from calls get assigned a dispatch group category (see Call Quality below). The dispatch group category of a job gets stored in the DISP_GROUP_CAT column of the JOBS table. This field is not available for display to the end user.

Column Name	Type	Description
name	VARCHAR2(31)	The name of the dispatch group category
description	VARCHAR2(31)	Any relevant description of the dispatch group category

DISPATCH_GROUPS

This table contains the set of valid dispatch groups. The dispatch group of a job gets stored in the REFERRAL_GROUP column of the JOBS table and can be displayed in Work Agenda by configuring the REFERRAL_GROUP column.

Column Name	Type	Description
name	VARCHAR2(31)	The name of the dispatch group
description	VARCHAR2(31)	Any relevant description of the dispatch group
is_default	VARCHAR2(1)	'Y' if this dispatch group is the default dispatch group or ‘N’ otherwise. Only one entry in this table should have ‘Y’

DISPATCH_GROUP_STRUCTURES

This table contains the set of mappings between dispatch group categories, control zones, and device classes to determine the associated dispatch groups. Matching against both device class and dispatch group category is mutually exclusive, so both these values are held in the column 'key'.

Column Name	Type	Description
control_zone	VARCHAR2(64)	A valid control zone name to match
key	VARCHAR2(31)	Either a valid device class name or a valid dispatch group category to match against the outage dispatch group category (if not empty) or device class (if the outage dispatch group category is empty)
dispatch_group	VARCHAR2(32)	The resultant dispatch group corresponding to a match of the above values.

Call Quality

Entries in the CALL_QUALITY configuration table with set_name=’dgroup_category’ are used to assign a dispatch group category to jobs created from calls. The trouble code of the first call on a job will be used to lookup the dispatch group category. If a match is found, the dispatch group category will be populated with the value from the CUST_PRIORITY column of the CALL_QUALITY table. If no match is found, the dispatch group category will default to "dispatch". When assigning the initial dispatch group to a job, device class based rules are checked before rules based on the dispatch group category.

Chapter 20

Damage Assessment Configuration

Schema

DAMAGE_REPAIR_TIMES

Column Name	Column Type	Description
ASSET_TYPE	VARCHAR2(32)	Damaged asset type. Either device class name (for modeled assets) or damaged part id (for non-modeled assets). For non-modeled assets the list of known part ids is configured in the DAMAGE_PARTS database table.
REPAIR_TIME_A	NUMBER	Number of minutes needed to repair single instance of this damage asset when the location is truck-accessible.
REPAIR_TIME_I	NUMBER	Number of minutes needed to repair single instance of this damage asset when the location is not truck-accessible.
DEFAILT_TQ	VARCHAR2(50)	Default trouble queue for this damage asset type.

Repair Times

Configuration for calculating repair time for different types of damaged assets is stored in the DAMAGE_REPAIR_TIMES table. This table contains repair minutes values for non-modeled asset types and device classes. For device classes inheritance is considered when applying configured values.

Default Work Queue

Configuration for automatically assigning work queues is stored in the DAMAGE_REPAIR_TIMES table. When damage assessment is created or updated trouble queues will be determined based on damaged asset and added to the event(s) associated with the damaged assessment.

Examples

Non-modeled damaged asset - tree. Assigned to work queue TRE.

```
INSERT INTO damage_repair_times (asset_type, repair_time_a,
                                 repair_time_i, default_tq)
VALUES ('40', 120, 120, 'TRE');
```

Device classes for modeled assets. Damaged lines assigned to work queue LNE.

```
INSERT INTO damage_repair_times (asset_type, repair_time_a,
                                 repair_time_i)
VALUES ('transformer', 180, 300);
INSERT INTO damage_repair_times (asset_type, repair_time_a,
                                 repair_time_i)
VALUES ('_fuse', 30, 30);
INSERT INTO damage_repair_times (asset_type, repair_time_a,
                                 repair_time_i)
VALUES ('wood_pole', 90, 180);
INSERT INTO damage_repair_times (asset_type, repair_time_a,
                                 repair_time_i, default_tq)
VALUES ('_electric_line_seg', 0, 0, 'LNE');
```

Damaged Asset Status Options

The possible values for the damaged asset status drop-down list are configured in the MESSAGE_CODE_LOOKUP database table. MESSAGE_TOPIC is 'DA_ASSET_STATUS'. Drop-down list items are sorted according to the values in the MESSAGE_ORDER column.

The MESSAGE_CODE 'OPEN' is used to indicate that the damaged device is open and customers are without power.

Example

```
INSERT INTO message_code_lookup (message_topic, message_code,
                                 message_string, message_order)
VALUES ('DA_ASSET_STATUS', 'NA', 'N/A', 1);
INSERT INTO message_code_lookup (message_topic, message_code,
                                 message_string, message_order)
VALUES ('DA_ASSET_STATUS', 'OPEN', 'Open', 2);
INSERT INTO message_code_lookup (message_topic, message_code,
                                 message_string, message_order)
VALUES ('DA_ASSET_STATUS', 'CLOSED', 'Closed', 3);
```

Damaged Asset Condition Options

The possible values for the damaged asset condition drop-down list are configured in the MESSAGE_CODE_LOOKUP database table. MESSAGE_TOPIC is 'DA_ASSET_CONDITION'. Drop-down list items are sorted according to the values in the MESSAGE_ORDER column.

Example

```
INSERT INTO message_code_lookup (message_topic, message_code,
                                 message_string, message_order)
VALUES ('DA_ASSET_CONDITION', 'OK', 'OK', 2);
INSERT INTO message_code_lookup (message_topic, message_code,
                                 message_string, message_order)
VALUES ('DA_ASSET_CONDITION', 'DAMAGED', 'Damaged', 1);
INSERT INTO message_code_lookup (message_topic, message_code,
                                 message_string, message_order)
VALUES ('DA_ASSET_CONDITION', 'DOWN', 'Down', 3);
```

'Road Blocked' Damaged Asset Type

The 'Road Blocked' damaged asset type indicates locations where roads are blocked. This asset type has part id 'RB'.

Chapter 21

Control Tool Configuration

The intended audience for this chapter are project engineers or software engineers responsible for configuring the Oracle Network Management System (NMS) Control Tool. This chapter includes the following topics:

- Overview
- Control Tool Configuration
  - Control Tool Database Table Configuration
  - The Control.xml File
  - Project_Control_Actions.inc Include File
- Updating Control Tool Configuration in Production Systems

Overview

The Control Tool affects many different aspects of the NMS system including tools, such as Web Switching and Web Safety, as well as services, such as DDService, PFService, and SwService. Due to the interactions with the various components, the Control Tool configuration includes database table configuration as well as JBot XML configuration typical of the other Java-based tools.

Control Tool Configuration

Control Tool Database Table Configuration

Control Tool actions are defined in two NMS database tables. The **CONTROL_ACT** table defines the actions and the **CONTROL_AGGREGATES** defines the order of execution for multistep control sequences.

The Control Tool Workbook generates the `CONTROL_ACT` and `CONTROL_AGGREGATES` table insert statements; the SQL statements can then be added to your `[project]_control.sql` file. You can find the workbook at:

```
$NMS_BASE/OPAL/workbooks/Oracle_Utilities_NMS_Control_Tool_1.11.xls
```

CONTROL_ACT Database Table Configuration

The CONTROL_ACT database table contains the definitions for each control action used in the Control Tool, as well as some actions used exclusively by Web Switching and Web Safety.

Definition

- act_key - a unique index for the record.
- act_cls - the action type for the action (see below)
- act_idx - the action identifier (see below)
- action_name - the name of the JBot Action to be executed for this record
- label - the label to display on the Control Tool
- instruct_label - the label for the instruct version of the action on the Control Tool, if desired
- switching_desc - the text displayed for the action in Web Switching steps
- switching_code - the short code used when entering manual steps in Web Switching
- description - the description displayed in Web Switching, User Log, and Event Log
- undo_act_key - the act_key of the undo action, used when creating go-back steps in Web Switching

Valid values for act_cls/act_idx pairs:

act_cls	act_idx	Description
CONDADD	tag, note, <condition name>	Add a condition of the passed type
CONDDLG	tag, note, <condition name>	Display the edit dialog of conditions of the passed type.
CONDREM	tag, note, <condition name>	Remove a condition of the passed type.
Commissioning	Action	A commissioning action (automatically added by the Commissioning Tool). Use the commissioning action in the action_name (WSW_STEP_COMMISSION, WSW_STEP_DECOMMISSION, WSW_STEP_UNDO_COMMISSION, WSW_STEP_UNDO_DECOMMISSION).
DDS	CLOSE, OPEN	Close or open the device.
DDS	EARTH, EARTH_DW	Place or remove an earth/ground on the device. If the device is a switch, the Control Tool will display a side selection dialog.
DDS	MOMENTARY	Create a momentary on the device.
FLISR	ISOLATE_RESTORE	A FLISR Isolate & Restore block (automatically added when creating a FLISR plan.
HLMsg	NOOP	Comment steps. Also used as the first step of an aggregate.

act_cls	act_idx	Description
JMS	<none>	An automatic JMService event step, used in the Event Log and User Log.
JMS	submit_calls, submit_amr_calls	Allows trainer to submit normal or AMI calls on or downstream from the selected device.
MTS	DISABLE_FLISR, ENABLE_FLISR	Disable or enable FLISR for the device.
Manual	NOOP	Manual steps.
NOOP	20, 30, <Number of seconds to wait>	Wait the specified number of steps. Only used by FLISR to wait for SCADA responses. The number of wait seconds is not supported outside of FLISR.
PFS	CREATE_FLA_FAULT, CREATE_FLISR_EVENT	Allows the user or trainer to trigger FLA and FLISR events with faults on the selected conductor.
SRS	PO_DOWN, PO_HERE, PO_UP	Move the outage downstream, to here, or upstream.
START	ControlEdit	Perform a model edit. See Model Edit Configuration on page B-1 for details.
Safety	Action	Safety actions. Use the action_name column to specify the action (issue, unissue, release, complete, abort)
ScadaCtrl	<1 and the attribute number, 2 and the attribute number>	Send a SCADA control for the passed digital attribute. Use 1 + the attribute to clear, and 2+ the attribute to set. For example, for attribute 3 (AutoReclose), you would set the act_idx to 13 to clear AutoReclose, and 23 to set AutoReclose.
Switching	Block	A Switching block. Use the action_name to specify the type of block (WSW_BLOCK_CONSTRUCTION, WSW_BLOCK_CUSTOM, WSW_BLOCK_DEFAULT, WSW_BLOCK_FAULT_LOCATION, WSW_BLOCK_ISOLATE, WSW_BLOCK_MAINTENANCE, WSW_BLOCK_NOMINAL, WSW_BLOCK_RESTORE)

act_cls	act_idx	Description
WCE	submit_call, submit_amr_call	Allows trainer to submit a normal or AMI call for a specific customer. The description column will be used as the resulting step's Details column. There are substitution parameters available for the WSW_STEP_CALL_ENTRY action: • $ACCOUNT_NAME: the customer name or address, if name is empty. • $CODE_NAMES: trouble code short descriptions, delimited by a dash. This is similar to the Work Agenda's Clues column. Empty for AMI calls. • $COMMENT: the call comment. Populated only for calls submitted using Web Call Entry.
WSW_BLOCK_SCADA_SHED	Block	SCADA block of shed steps in Load Shed sheet.
WSW_BLOCK_SCADA_RESTORE	Block	SCADA block of restoration steps in Load Shed sheet

CONTROL_AGGREGATES Database Table Configuration

The CONTROL_AGGREGATES table lists the aggregate Control Tool actions in the order they are to be executed.

> **Note:** aggregates should not contain any SCADA actions because SCADA responses return to the NMS one at a time. As such, they need to be configured as separate actions and separate switching steps.

Definition

- parent_act_key – The first act_key, from the CONTROL_ACT table
- act_key – The nth act_key, also from the CONTROL_ACT table
- sequence_number – The order of the action

For example, if you had CONTROL_ACT records for:

- 100: HLMsg::NOOP, "Open & Tag"
- 200: DDS::OPEN, "Open"
- 300: CONDADD:tag, "Place Tag"

Then you would enter the following rows in your CONTROL_ACGGREGATES table:

- parent_act_key=100, act_key=200, sequence_number = 1
- parent_act_key=100, act_key=300, sequence_number = 2

CONTROL_ACT_PROMPTS Database Table Configuration

This table holds the attributes that are presented to the user when creating or editing a model edit. Model edit CONTROL_ACT records hold all their configuration parameters in the DESCRIPTION field. This field can contain user-editable parameters that are defined by $PROMPT(<name>). For example, the **P-P Cross Phase Prompt...** is defined in the CONTROL_ACT table as follows:

```
INSERT INTO control_act
  (act_key, act_cls, act_idx, action_name, label, instruct_label,
   switching_desc, switching_code, description, undo_act_key)
VALUES
  (451, 'START', 'ControlEdit', 'ACT_PPCROSS', 'P-P Cross Phase
Prompt...', '',
   '', '', '-jumper -status 0 -nominal 0 -deviceCls 447 -gapLength 20 -
crossPhase -handle $DEVICE -alias $PROMPT(JmpAlias) -cesuser $CESUSER
-x $WX -y $WY -symbolCls 44707 -coordSys $COORDSYS', 0);
```

This means that the user is presented with a "JmpAlias" prompt when creating the jumper, so this JmpAlias needs to be defined in the CONTROL_ACT_PROMPTS table:

```
INSERT INTO control_act_prompts
  (prompt_key, prompt_string, required, data_type, default_value,
list_values)
VALUES
  ('JmpAlias','Jumper Alias','Y','string','Jumper_$WX_$WY','');
```

These prompts are also used when editing model edit attributes. In this case, the prompt_key needs to match the column in the attributes view for that device class. See Model Edit Configuration on page B-1 for details.

The Control.xml File

Once you have defined the control actions, you need to specify which buttons to appear on the Control Tool for the device classes. You also need to map these buttons to the control actions that were defined in the CONTROL_ACT table, and you need to create JBot actions to match the `CONTROL_ACT.action_name` values.

Set up your `<Button>` or `<PopupMenuItem>` element like any other JBot button, but with a few important differences:

- Use the `data_source` attribute to list "`DS_LABELS.<the button name>`" or "`DS_LABELS.<the button name>:INSTRUCT`" to use the CONTROL_ACT.label or CONTROL_ACT.instruct_label.
- Set the <Visible> element based on the inheritance or the device class itself. It is recommended that you set up parent classes in your [project]_inheritance.dat for each logical grouping of device classes that will have different Control Tool options, then use those in these "`when`" clauses. For example:

```
<Visible initial="false"
when="{DS_CONTROL_TOOL.DEVICE_CLASS_PARENTS ==
'control_tool_switch'}"/>

<Visible initial="false"
when="{DS_CONTROL_TOOL.DEVICE_CLASS_PARENTS ==
'control_tool_breaker'}"/>

<Visible initial="false" when="{DS_CONTROL_TOOL.DEVICE_CLASS in
('generator')}"/>
```

- Add <ControlActions> and <ControlAction> elements to list the CONTROL_ACT keys to use for each device class or group of device classes. List the actions in order and use the "when" clause so the Control Tool knows which CONTROL_ACT record you want to use for each device class. The first matching CONTROL_ACT key will be used. You can configure a default at the end of the list with no "when" clause to catch all others that do not match earlier keys. For example, you may configure different actions and button labels for an Open button (*Disconnect Generator*, *Disconnect Jumper*, *Open Switch*, etc.):

```
<ControlActions>
    <ControlAction key="170" when="{DS_CONTROL_TOOL.DEVICE_CLASS in
    ('generator')}"/>
    <ControlAction key="210" when="{DS_CONTROL_TOOL.DEVICE_CLASS in
    ('inline_jumper','p_p_jumper','rack_sub_jumper','sub_jumper')}"/>
    <ControlAction key="580" when="{DS_CONTROL_TOOL.DEVICE_CLASS_PARENTS ==
    'switch'}"/>
</ControlActions>
```

- List the JBot actions to perform in the <PressPerform> element. For operations and other actions you record in switching, you should always add an ACT_BEGIN_ACTION and an ACT_END_ACTION call to set flags, reset the control tool, and prepare it for the next user action.

  Note: buttons that only display other tools do not need the ACT_BEGIN_ACTION and ACT_END_ACTION actions.

  Pass the $INSTRUCT_FLAG$ to the ACT_BEGIN_ACTION as true if this is an instruct button. Pass the $SEND_TO_SWITCHING$ flag to the ACT_END_ACTION if this actions should be recorded in Switching or the Misc Log.

- List the JBot action you configured as the CONTROL_ACT.action_name between the ACT_BEGIN_ACTION and ACT_END_ACTION actions. For example:

```
<PressPerform>
    <Command value="ExecuteActionCommand">
        <Config name="action" value="ACT_BEGIN_ACTION"/>
        <Config name="$INSTRUCT_FLAG$" value="true"/>
    </Command>

    <Command value="ExecuteActionCommand">
        <Config name="action" value="ACT_OPEN"/>
    </Command>

    <Command value="ExecuteActionCommand">
        <Config name="action" value="ACT_END_ACTION"/>
        <Config name="$SEND_TO_SWITCHING$" value="true"/>
    </Command>
</PressPerform>
```

Example

```
<PopupMenuItem name="BTN_INSTRUCT_OPEN_DEVICE"
class="javax.swing.JMenuItem"
data_source="DS_LABELS.BTN_INSTRUCT_OPEN_DEVICE:INSTRUCT">
  <Enabled initial="false" when="OPEN_DEVICE and
(DS_CONTROL_DEFAULT.CURRENT_MODE == 'RT')"/>
  <Visible initial="false" when="{DS_CONTROL_TOOL.DEVICE_CLASS_PARENTS
== 'switch'}"/>
  <ControlActions>
     <ControlAction key="170" when="{DS_CONTROL_TOOL.DEVICE_CLASS in
('generator')}"/>
```

```
    <ControlAction key="210" when="{DS_CONTROL_TOOL.DEVICE_CLASS in
('inline_jumper','p_p_jumper','rack_sub_jumper','sub_jumper')}"/>
    <ControlAction key="580"
when="{DS_CONTROL_TOOL.DEVICE_CLASS_PARENTS == 'switch'}"/>
  </ControlActions>
  <ValidValues>
    <RunGroup run_group="CHECK_OPERATION_TIME"/>
  </ValidValues>
  <PressPerform>
      <Command value="ExecuteActionCommand">
          <Config name="action" value="ACT_BEGIN_ACTION"/>
          <Config name="$INSTRUCT_FLAG$" value="true"/>
      </Command>
      <Command value="ExecuteActionCommand">
          <Config name="action" value="ACT_OPEN"/>
      </Command>
      <Command value="ExecuteActionCommand">
          <Config name="action" value="ACT_END_ACTION"/>
          <Config name="$SEND_TO_SWITCHING$" value="true"/>
      </Command>
  </PressPerform>
</PopupMenuItem>
```

Mapping CONTROL_ACT Database Records to JBot Configuration

The Control.xml file defines a list of buttons and a <ControlAction> to perform when each button is pressed. The <ControlAction> "key" attribute should match the CONTROL_ACT.act_key in the database.

In the <PressPerform> element, the button should then prepare for an atomic action by calling the "ACT_BEGIN_ACTION", the specific action (as defined in a CONTROL_ACTIONS include file), then the "ACT_END_ACTION". In general, the button's specific action should match the CONTROL_ACT.action_name. All actions that are executed in the <PressPerform> element need to be defined in a CONTROL_ACTIONS include file.

When the recorded switching step is later instructed or executed in Web Switching, the CONTROL_ACT.action_name for this key is used to determine the correct action (from the CONTROL_ACTIONS include file) to execute. Any CONTROL_ACT.action_name actions that are not defined in a CONTROL_ACTIONS include file will error when attempting to instruct or execute the step in Web Switching

Commonly Used Flags and Datastore Values

Commonly used flags and datastore values that can be used in when clauses:

- `OPEN_DEVICE`/`CLOSE_DEVICE`: open/close are valid actions based on the state of the device.
- `DS_CONTROL_DEFAULT.CURRENT_MODE`: *RT* (*i.e.*, real-time) or *STUDY*.
- `DS_CONTROL_TOOL.DEVICE_CLASS_PARENTS`: a set of all parent classes for the selected device.
- `DS_CONTROL_TOOL.DEVICE_CLASS`: the selected device class name.
- `HAS_<`*`condition class name`*`>`: whether a condition of the class (capitalized) is active on the device. (For example, `HAS_INSTRUCT`, `HAS_TAG`, `HAS_NOTE`, etc.).
- `HAS_<`*`condition`*`>_<`*`status 0-10`*`>`: whether a condition with status 0-10 is active on the device. (Example: `HAS_INFO_0`, `HAS_HOLD_2`, etc.).
- `SCADA_OPERATED`: if there is SCADA telemetry on the device status point.

- HAS_MEAS_<SCADA measurement>: whether the device has a SCADA measurement (analog or digital) with the name (capitalized). (For example, HAS_AUTORECLOSE, HAS_AMPS, etc.)
- <SCADA measurement>_ON/<SCADA measurement>_OFF: whether the digital measurement is ON or OFF. (For example, AUTORECLOSE_ON, AUTORECLOSE_OFF, FAULT_INDICATOR_ON, etc.)
- DS.CONTROL_TOOL.QUALITY_<scada measurement>: datastore value is set to the quality string of the measurement. For example, QUALITY_STATUS might be blank (no quality), QUALITY_AUTO_RECLOSE might be *M* (manual), and QUALITY_AMPS might be *IM* (manual and inhibited).
- DS.CONTROL_TOOL.QUALITY_<scada measurement>_BITS: datastore value is set to a bitwise integer containing all the quality bits.
- DS_LOGIN_ENTRY.WEB_SWITCHING_ENABLED: *true* or *false*.
- FROM_SWITCHING: whether the action is being instructed or completed from Web Switching.
- HIDE_CONTROL_TOOL: option
- INSTRUCT_ONLY: whether the action being executed is an *Instruct* (as opposed to a *Complete*).
- IS_OPENED: the device is opened on all phases.
- IS_CLOSED: the device is closed on all phases.
- IS_ISOLATED: the device is marked as an isolation point.
- IS_SECURED: the device is marked as a secure point.
- IS_IGNORED: the device is ignored, either because it is pending construction or is decommissioned.

Optional Project Status Flags

- HIDE_CONTROL_TOOL - if set implemented in the project Control.xml and set to 'true,' this flag will close the Control Tool when another device is selected in the Viewer. If set to false, the Control Tool to remain open. This option is not included in the product version of Control.xml.

  To implement this option, add the following (with or without the comment at the beginning of the code snippet) to the project Control.xml file:

```
<!-- Control Tool window behavior. This is a configuration
option. Set this to 'true' to enable closing of Control Tool
when another device is selected in the Viewer. Set to false if
you want the Control Tool to remain open. -->

<Perform name="Window" category="windowOpened">
  <Command value="SetStatusFlagCommand">
    <Config name="flag_names" value="HIDE_CONTROL_TOOL"/>
    <Config name="flag_values" value="true"/>
  </Command>
</Perform>
```

Project_Control_Actions.inc Include File

The product **Control.xml** file includes the **CONTROL_ACTIONS.inc** file, which contains all of the product **<Action>** definitions. Project-specific actions should be defined in a **PROJECT_CONTROL_ACTIONS.inc** file.

> **Note:** You may find it useful to use the **CONTROL_ACTIONS.inc** as an example.

The following example illustrates how to define an action to add an Information tag. The condition class, **info**, is defined in the `CLASSES` table.

```
<Action name="ACT_ADD_INFO">
    <Command value="ExecuteActionCommand">
        <Config name="action" value="ACT_ADD_CONDITION"/>
        <Config name="$CONDITION_CLASS$" value="info"/>
    </Command>
</Action>
```

Web Switching executes actions when you instruct or complete steps in Web Switching; therefore, if there is any validation needed to prevent execution or completion of steps based on device states, you should add it to the `<Action>` element, using *DisplayErrorCommand*. You may add any number of specific error messages.

If you wanted, for example, to enforce that the system can only place an informational tag on a device that has no active instructs, then you could add the following:

```
<Action name="ACT_ADD_INFO">
    <Command value="DisplayErrorCommand" when="HAS_INSTRUCT">
        <Config name="message_code" value="CANNOT_HAVE_INSTRUCT"/>
    </Command>
    <Command value="ExecuteActionCommand">
        <Config name="action" value="ACT_ADD_CONDITION"/>
        <Config name="$CONDITION_CLASS$" value="info"/>
    </Command>
</Action>
```

And in MessageCode_en_US.properties, you would need the following:

```
CANNOT_HAVE_INSTRUCT=Cannot perform this action when an instruct
is present.
CANNOT_HAVE_INSTRUCT.title=Action Failed
```

Or you might only want to perform that check for instructs if the user is not instructing the current action:

```
<Action name="ACT_ADD_INFO">
    <Command value="DisplayErrorCommand" when="!INSTRUCT_ONLY
      and HAS_INSTRUCT">
        <Config name="message_code" value="CANNOT_HAVE_INSTRUCT"/>
     </Command>
    <Command value="ExecuteActionCommand">
        <Config name="action" value="ACT_ADD_CONDITION"/>
        <Config name="$CONDITION_CLASS$" value="info"/>
    </Command>
</Action>
```

And if, for example, you also wanted to make sure the tool is in study mode, you could add another specialized message, either before or after the other message:

```
<Action name="ACT_ADD_INFO">
    <Command value="DisplayErrorCommand" when="!INSTRUCT_ONLY and
HAS_INSTRUCT">
        <Config name="message_code" value="CANNOT_HAVE_INSTRUCT"/>
     </Command>
```

```
    <Command value="DisplayErrorCommand"
      when="DS_CONTROL_DEFAULT.CURRENT_MODE == 'RT'">
        <Config name="message_code" value="CANNOT_USE_RT"/>
    </Command>
    <Command value="ExecuteActionCommand">
        <Config name="action" value="ACT_ADD_CONDITION"/>
        <Config name="$CONDITION_CLASS$" value="info"/>
    </Command>
</Action>
and:
CANNOT_USE_RT=Cannot perform this action in real-time mode.
CANNOT_USE_RT.title=Action Failed
```

Using the Control Config Generator

The NMS Control Config Generator is a stand alone application that is used to read the Control Tool workbook, parse the contents, and generate the desired configuration files.

Below is a list of all files that can be generated with a brief description. Notice that `[project]` indicates that the files generated pertain to a specific project configuration.

Generating Data Files

1. Make the appropriate changes to the Control Tool workbook.
2. After the changes are made, open the Control Config Generator Tool from the command line by using one of the following methods:
    a. On Unix, launch the generator using the following script:

    ```
    $NMS_BASE/bin/control-config-generator
    ```

    b. To run on a Microsoft Windows PC, your first need to zip the `NMS_BASE/bin/ControlConfigGenerator` directory by running the following command on a Unix terminal:

    ```
    zip -ry ControlConfigGenerator.zip $NMS_BASE/bin/
    ControlConfigGenerator
    ```

    Then move the zip file to the PC, unzip the file, and then navigate to the ControlConfigGenerator directory, and double-click the ModelConfigGenerator.jar file to launch it (assuming java is installed).
3. Enter the project name, if it is not populated.
4. Browse to the location of the modified Control Tool Workbook.

    The tool will read the workbook into memory.
5. Browse to a location where you want to save each of the configuration files.
6. Click **Generate** to generate the configuration files.

    The Control Tool Generator Tool will generate the desired configuration files and create reports for any errors encountered.
7. The resulting files should be manually merged into the standard configuration files, as described above.

Configuration Example: Adding an Undo Close Action

Adding the Undo Close function to the Control Tool requires modifications to the project CONTROL_ACTIONS.inc, Control.xml, and Control_en_US.properties files.

CONTROL_ACTIONS.inc

Add the ACT_UNDO_CLOSE action:

```
<Action name="ACT_UNDO_CLOSE">
    <Command value="OpenDeviceCommand">
        <Config name="work_deenergized" value="false"/>
        <Config name="trace_direction" value="2"/>
        <Config name="enable_dialog" value="true"/>
        <Config name="must_view_abnormals" value="false"/>
        <Config name="must_view_atos" value="true"/>
        <Config name="must_view_conditions" value="true"/>
        <Config name="show_breaker_info" value="true"/>
        <Config name="breaker_info_fields"
         value="FEEDER_ID_1,NOMINAL_VOLTAGE, INTERRUPTION_RATING"/>
        <Config name="no_phases" value="false"/>
        <Config name="undo_close" value="true"/>
    </Command>
    <Command value="ExecuteActionCommand">
        <Config name="action" value="OPERATE_DEVICE"/>
    </Command>
</Action>
```

Control.xml

Add the button/menu item to Control.xml.

> **Note:** the value of control action key will depend on the project's configuration.

```
<PopupMenuItem name="BTN_UNDO_CLOSE"
class="javax.swing.JMenuItem"data_source="DS_LABELS.BTN_UNDO_CLOSE">
    <Enabled initial="false" when="OPEN_DEVICE and !HAS_INSTRUCT"/>
    <Visible initial="false"
     when="{DS_CONTROL_TOOL.DEVICE_CLASS_PARENTS == 'switch'}"/>
    <ControlActions>
        <ControlAction key="6480"
         when="{DS_CONTROL_TOOL.DEVICE_CLASS_PARENTS == 'switch'}"/>
    </ControlActions>
    <ValidValues>
        <RunGroup run_group="CHECK_OPERATION_TIME"/>
    </ValidValues>
    <PressPerform>
        <Command value="ExecuteActionCommand">
          <Config name="action" value="ACT_BEGIN_ACTION"/>
          <Config name="$INSTRUCT_FLAG$" value="false"/>
        </Command>
        <Command value="ExecuteActionCommand">
          <Config name="action" value="ACT_UNDO_CLOSE"/>
        </Command>
        <Command value="ExecuteActionCommand">
          <Config name="action" value="ACT_END_ACTION"/>
          <Config name="$SEND_TO_SWITCHING$" value="false"/>
        </Command>
    </PressPerform>
</PopupMenuItem>
```

Control_en_US.properties

Add label and tooltip for the new button/menu item.

```
BTN_UNDO_CLOSE.text = Undo Close
BTN_UNDO_CLOSE.tooltip = Undo latest close operation for the selected
switch
```

Control Tool Options in the Viewer Context Menu

Control Tool options in the Viewer's context menu are configured in the VIEWER_POPUP.inc file. The context menu options reference Control.xml buttons and automatically use their **<Visible>** and **<Enabled>** elements.

When you right-click in the Viewer, a call is configured to the **ViewerRightClickCommand**. This command populates the Control Tool data stores; based upon the visibility and enabled property of different buttons and menu items, it will make the Control Tool menu items visible in the Viewer's context menu. These Viewer menu items need to have names that match the Control Tool buttons, but with **MNU_ prefixes** instead of **BTN_ prefixes**.

You can optionally pass the **menu_names** parameter to the **ViewerRightClickCommand** command. The **menu_names** parameter lists the context menu names to consider when looking for matching Control Tool buttons. **OPERATE_POPUP** is included by default. Include any additional context menus and separate them with commas.

Updating Control Tool Configuration in Production Systems

After a system is in production, Control Tool updates are typically applied for one or more of the following reasons:

- to provide control actions for new device classes that are being added to the network (see Adding New Device Classes).
- to map existing actions to existing device classes (see Mapping Existing Actions to Existing Device Classes).
- to add new actions (for existing device classes; see Adding New Actions).
- to change when an action is enabled (see Changing When Actions are Enabled).

Adding New Device Classes

Configure the new classes in your [project]_inheritance.dat to inherit from the correct superclass so that it automatically gets the desired Control Tool buttons.

Mapping Existing Actions to Existing Device Classes

Either change the inheritance of certain device classes to get the desired set of buttons, or change the <Visible> and the <ControlAction> elements in the buttons to include the added superclass.

Adding New Actions

Add a new CONTROL_ACT record and reference a new JBot Action name in it.Then use the existing PROJECTS_ACTIONS.inc examples as a guide and add the new JBot Action to it.

Also create the button in the Control.xml file, reference the CONTROL_ACT.act_key in it, and set up the <Visible> and <Enabled> tags.

Changing When Actions are Enabled

Modify the <Enabled> tag in the Control.xml file, as with the other JBot tools.

Aggregate, Secondary, or Associated Devices

If you have the AGGREGATE_DEVICES model table populated and you wish to operate devices from a different device's Control Tool, you will need to make use of the UseAggregateModelDeviceCommand and UsePrimaryDeviceCommand. Use the UseAggregateModelDeviceCommand and pass it the index of the aggregate to use and the subsequent operation Commands will operate the aggregate device. Use the UsePrimaryDeviceCommand to move control back to the selected device if you are performing aggregate actions.

Note that the Look Ahead does not take into account any previously instructed actions. So when you instruct an aggregate action on multiple devices, the Look Ahead results will not reflect the results of any previous actions.

Customer Counts Lists in the Look Ahead

The Drop Count, Pickup Count, and ATO Customers tabs in the Look Ahead can display four standard rows, plus any number of grouped critical customer type rows.

- NUMCUST: Total Customers
- CRITCUST: Critical Customers
- DERKVA: DER kVA
- REVENUE: Total Revenue (Not used in standard product configuration)

The "critCustGroup" SRS Rule allows you to configure groupings of different critical customer types, which are also available in these tables. For example, product configuration contains groups for **KEY**, **EMER**, and **MED**, so **CRIT_KEY**, **CRIT_EMER**, and **CRIT_MED** are also available as rows.

Define the row labels using the LOOKAHEAD_ROW_NAME JBotFormat, and configure the order using the LOOKAHEAD_ROW_NUM. Hide unwanted rows using the SetFilterCommand in your DLG_LOOKAHEAD.xml.

Configurable Device Lists in the Look Ahead

Product is configured to display a "**View Cap. and Reg.**" table. This set of devices is configured using the "lookahead_dev_list" abstract base class. However, you can configure any set of device classes in this tab using this inheritance scheme, and you can rename the tab to match.

Chapter 22

Web Switching Management Configuration

This chapter describes how to configure and administer Web Switching Management. It includes the following topics:

- Configuring Classes and Inheritance
- Database Data Tables
- Database Configuration Tables
- Configuring Project-Specific Columns
- Global Web Switching Parameters
- GUI Configuration Overview
- Switching Sheets
- Switching Steps
- Web Safety
- High Level Messages
- Troubleshooting

Configuring Classes and Inheritance

Web Switching Management utilizes standard classes to define the switching sheet types. The following table lists the classes utilized by Web Switching Management:

Class Name	Purpose
switch_sheet_step	The class is used for switching step handles. This class is defined as part of the core classes and should not be changed.
study_switch_sheet_step	The class is used for study time step state transitions. The class allows us to configure different transitions for steps in study mode. This class is defined as part of the core classes and should not be changed.
switch_sheet_planned	The sheet class used for Planned switching sheet handles. This class is defined as part of the core classes and should not be changed.
switch_sheet_emergency	The sheet class used for Emergency switching sheet handles. This class is defined as part of the core classes and should not be changed.
switch_sheet_fault	This sheet class is not used by Product configuration, but it is defined as part of the core classes. This class can be redefined and given a new name if the project wants a new switching sheet type. The switching sheet class numbers are referenced in the SWMAN_SHEET_CLS database configuration table.
oc_switch_sheet	The sheet class used for Outage Correction switching sheet handles. This class is defined as part of the core classes and should not be changed.
flisr_switch_sheet	The sheet class used for FLISR switching sheet handles. This class is defined as part of the core classes and should not be changed.
voltvar_switch_sheet	The sheet class used for VVO switching sheet handles. This class is defined as part of the core classes and should not be changed.
switch_template	The sheet class used for Template switching sheet handles. This class is defined as part of the core classes and should not be changed.
switch_sheet_training	The sheet class used for Training scenario switching sheet handles. This class is defined as part of the core classes and should not be changed.
loadshed_switch_sheet	The sheet class used for Load Shed switching sheet handles. This class is defined as part of core classes and should not be changed.

Class Name	Purpose
switch_sheet	This class is used to give the Planned, Emergency, FLISR and Outage Correction sheet types their unique switching sheet numbers. The next_free_index value for this class in the CLASSES table defines the next available sheet number to use for these four sheet types. Since all four of these sheet types gather their switching sheet numbers from the same pool, none of them can have identical sheet numbers. For more information, see Sheet Types. For Product configuration, this class also is set up to inherit from classes switch_sheet_planned, switch_sheet_emergency and switch_sheet_fault. This inheritance defines whether events are associated to the steps recorded into the sheets. Events are associated to steps so that events follow the steps if they are moved from one sheet to another. If you define a new Planned or Emergency sheet type for your project, then you will need to add that new class to the list of classes that the switch_sheet class inherits from.
switch_misc	The sheet class used for the Miscellaneous Log handle. This class is defined as part of the core classes and should not be changed.
switch_sheet_safety	The sheet class used for stand alone safety documents. Stand alone safety documents are sheets behind the scenes with a web safety GUI front end. This class also triggers special processing throughout the system to process stand alone safety documents correctly. This class is defined as part of the core classes and should not be changed.
ss_isolate	This class inherits from the list of device class types that should be used to generate isolation steps for the Generate Isolate Steps button option found on the conductor based Control Tools. When looking for isolation points and a device of the inherited class type is found, then switching steps to open and tag the device will be generated. For more information, see Generate Isolation Steps.
ss_secure	This class inherits from the list of device class types that should be used to generate tagging switching steps for devices that are already open. When selecting the Generate Isolate Steps option on the conductor based Control Tool and a device of the inherited device class is traced to and found open, then it will be tagged. For more information, see Generate Isolation Steps.
safety_num_INFO	The safety document class used for INFO safety document handles. The safety document classes are referenced in the SWMAN_SAFETY_TYPES database configuration table.
safety_num_CLEAR	The safety document class used for CLEAR safety document handles.
safety_num_HOLD	The safety document class used for HOLD safety document handles.
safety_num_HOT	The safety document class used for HOT safety document handles.

Class Name	Purpose
safety_num_WARN	The safety document class used for WARN safety document handles.
safety_num_DCZ	The safety document class used for Delegated Control safety document handles.
safetytag	This class inherits from the safety condition classes and defines whether a condition in status 1 to 999 can be removed. If a condition is a descendant of this safetytag class, then it cannot be removed until its status is returned to 0 or its status is greater than or equal to 1000.

Database Data Tables

The following database data tables are used by Web Switching Management to store data related to switching sheets and safety documents. Most of the tables used by Web Switching Management should not require any changes by the project. There are however a few tables that will require changes if the project adds additional data elements to the switching sheet body, the switching steps or the safety document body. Those tables are SWMAN_SHEET, SWMAN_SHEET_HIST, SWMAN_STEP and SWMAN_SAFETY_DOCS. All the other Web Switching and Web Safety tables will not require any changes unless a database table field size needs to be increased because of project specific data requirements.

SWMAN_AUDIT_LOG. This table stores all the audit log entries for the switching sheets and safety documents. Safety documents also get their audit log entries from the SWMAN_STEP table. This would include when conditions are applied and removed for a document.

SWMAN_CONTRACTOR_CREWS. This table stores all the contractor crews that have been created for Web Switching or Web Safety. These contractor crews are not visible or available to any other NMS tools in the system. The crews are created by filling out the form on the Select Crew and Select Safety Crew dialogs at the top of the Contractors tab. The table has a couple of custom text fields called CUSTOM_1 and CUSTOM_2 that can be used by the project to add additional fields and content to the contractor crew records. The Select Crew dialogs will need to be altered by the project to include the custom fields.

SWMAN_CREWS. This table stores the user assigned crews to the sheets. These crews are assigned to the sheet using the Select Crew dialog. It is possible to also add records to this table and define default crews that should show up on every switching sheet's Crews list. See Default Crews on page 22-33 for more information.

SWMAN_DELETED_CUSTOMER. This table stores a list of customers that were deleted from a sheet's impacted customer list. These customers are not actually deleted from the model. They are just marked as being removed from the impacted customer list.

SWMAN_IMPACTED_SUPPLY_NODES. This table stores the list of supply nodes impacted by a switching sheet's steps. In most cases, this list is generated manually by a user and is generated against the user's Study session.

SWMAN_PATCHES. This table will normally only ever have one record and that's the last model edit or build patch that was processed by the Web Switching service. The service determines the devices affected by the patch listed and flags any steps related to those devices. This table is used by internal processing and does not contain any data that may be displayed to a user.

SWMAN_SAFETY_CREWS. This table stores the user assigned crews to the safety documents. These crews are assigned to the safety document using the Select Safety Crew dialog. These are the crews that show up in the Assigned Crews list on the safety documents.

SWMAN_SAFETY_DOCS. This is the core data table for all the safety documents. This data table includes all the core information about the safety document like what state it is in, what sheet it is associated to, the crew it was issued to and whether it had been deleted or not. Additional project-specific columns can be added to this table. The mapping for these columns is configured in the eclipselink-orm.xml file. See Configuring Project-Specific Columns on page 22-7 for more information.

SWMAN_SHEET. This is the core data table for all the switching sheets. This data table includes all the core information about the switching sheet like what state it is in, the sheet's version, the master device associated to the sheet, Start and Finish dates and other key elements pertaining to the sheet. The general rule is that if any value on the switching sheet has any code based processing, then it gets included in this table. Additional project-specific columns can be added to this table. The mapping for these columns is configured in the eclipselink-orm.xml file. See Configuring Project-Specific Columns on page 22-7 for more information.

> **Note:** Any changes made to this table have to also be made to the SWMAN_SHEET_HISTORY table.

SWMAN_SHEET_DOCUMENTS. This table stores all the external documents that have attached to the switching sheet. The documents are stored as BLOBs in this table. The table also includes a user description about the attachment, the file name and the size of the file.

SWMAN_SHEET_HIST. This table stores a copy of the current sheet just before its version is incremented. This table is used to determine the differences between two switching sheet versions. Currently, there is no mechanism in place to display these differences to the user on the GUI. This table is being populated for reporting and diagnosis purposes only. Additional project-specific columns can be added to this table. The mapping for these columns is configured in the eclipselink-orm.xml file. See Configuring Project-Specific Columns on page 22-7 for more information.

> **Note:** Any changes made to this table have to also be made to the SWMAN_SHEET table.

SWMAN_SHEET_VIEW_AREA. This table maintains the list of view areas that have been created and associated to each of the switching sheets.

SWMAN_STEP. This is the core data table for all the switching sheet steps. This data table includes all the core information about the switching sheet steps like what state the step is in, the sheet version the step was added under, the device associated to the step, and other key elements pertaining to the steps. The general rule is that if any value within the step has any code based processing, then it gets included in this table. Additional project-specific columns can be added to this table. The mapping for these columns is configured in the eclipselink-orm.xml file. See Configuring Project-Specific Columns on page 22-7 for more information.

swman_audit_log_crews. This table stores the crews to the switching sheet audit log entries. Some crew related audit log entries can be associated to multiple crews, so this table is used to manage that list of crews. These are the crews that show up in the Crews field of the audit log.

swman_safety_edit_log. This table stores the edit log entries for a safety documents. Configured fields within a safety document can be logged as edits are made to them. This table stores those edits and is used to populate the safety document's Edit Log.

switch_sheet_outage_events. These are the manual and soft sheet to outage event associations. The table has a sheet handle and an event handle. JMService manages this list.

SWMAN_STEP_CREWS. This table stores the user assigned crews to the switching sheet steps. These crews are assigned to the switching steps when selecting steps and crews during the instructing and completing of switching steps. These are the crews that show up in the Instructed To field of the switching step.

Database Configuration Tables

The following database configuration tables are used by Web Switching Management to store configuration settings related to switching sheets and safety documents.

SWMAN_EVENT_ASSOC_TYPE. This table maps the event association types to names. Projects should only change these records if they wish to change the names of the associations.

SWMAN_SAFETY_TYPE_ACTIONS. This table maps the various types of step actions that can be associated to each safety document type. For instance CONDADD/hold actions can only be associated to HOLD documents, which DDS/OPEN operations can be associated to HOLD, CLEAR and HOT safety documents. This table controls these association rules.

SWMAN_SAFETY_TYPES. This table defines all the different types of safety documents configured for a project. Product configuration includes HOLD, Clearance, Informational, HOT, Delegated Control, and Warning safety document types. This table defines the following for each safety document type:

- The JBot tool panel and dialog that should be displayed when loading a safety document of this type.
- The numbering pool the safety document should use when generating unique document numbers. Product is configured to have each document type get their unique document numbers from separate numbering pools.
- The short description of each safety document type.

SWMAN_SHEET_CATEGORY. This table defines the list of sheet categories that every sheet type has to inherit from. Projects should not have any reason to alter the records in this table as they are pre-defined. The table should only be used to look up the description for each of the sheet categories. For instance, all sheets of category PLANNED will generate planned events when completing switching steps that impact customers. Each of the categories has pre-defined rules that define how the switching sheets should behave.

SWMAN_SHEET_CLS. This table defines the types of switching sheets configured for a project. This table defines the following for each switching sheet type:

- The sheet category the sheet type should inherit from.
- The JBot tool panel that should be displayed when loading a sheet of this type.
- The display order of the sheet types in the New Switching Sheet dialog.

- The numbering pool the sheet should use when generating unique sheet numbers. For instance Planned and Emergency sheets get their sheet numbers from the same numbering pool.
- The description of each sheet type. This description is displayed on the New Switching Sheet dialog.

SWMAN_STEP_STATE_MAPPING. This table is used by FLISR to map step state keys to a value of 0, 1, 2, or 16 where:

- **0** indicates that the step has no state
- **1** refers to any step in a completed state
- **2** is in reference to instructed states.
- **16** indicates that the step is deleted.

Configuring Project-Specific Columns

If your project added custom data fields to the SWMAN_SHEET, SWMAN_SHEET_HIST, SWMAN_STEP OR SWMAN_SAFETY_DOCS tables, then additional configuration is required to get Web Switching and Web Safety to utilize those new fields. Web Switching and Web Safety use EclipseLink to manage writing and reading of data to and from the database. For this package to work, a mapping has to be defined for each Java class that accesses the previously mentioned tables. This mapping for the most part is defined in the code, but any additional mappings have to be defined in the eclipselink-orm.xml file. The below set of directions should be used to define your project version of this file.

1. In your [project]/jconfig directory, create a subdirectory structure as follows:

   `[project]/jconfig/override/fwserver.jar/META-INF/`

2. Copy the Product version of the `eclipselink-orm.xml` file used by the switching and safety classes.

   Note: This file can be found in the product/jconfig/override/fwserver.jar/META-INF/ directory.

3. Save the file to the META-INF directory that you created in step 1.
4. You should find the following entries:

```
<entity class="com.splwg.oms.model.entity.swman.SwmanSheet">
<entity class="com.splwg.oms.model.entity.swman.SwmanSheetView">
<entity class="com.splwg.oms.model.entity.swman.SwmanStep">
<entity class="com.splwg.oms.model.entity.swman.SwmanStepView">
<entity class="com.splwg.oms.model.entity.safety.SafetyDocument">
```

   Each `<entity>` should contain an `<attributes>` Element with multiple `<basic>` elements that list the attribute name used in the configuration as well as the database column name.

   For example:

```
<entity-mappings xmlns="http://www.eclipse.org/eclipselink/xsds/persistence/
orm" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.eclipse.org/eclipselink/xsds/persistence/orm
http://www.eclipse.org/eclipselink/xsds/eclipselink_orm_2_1.xsd"
version="2.1">
  <entity class="com.splwg.oms.model.entity.swman.SwmanSheet">
    <attributes>
    <basic name="AlternateFinishDate" access="VIRTUAL" attribute-type="Date">
    <column name="ALTERNATEFINISHDATE"/>
```

```
    <temporal>TIMESTAMP</temporal>
    </basic>
    <basic name="AlternateStartDate" access="VIRTUAL" attribute-type="String">
    <column name="ALTERNATESTARTDATE"/>
    <temporal>TIMESTAMP</temporal>
    </basic>
    <basic name="ChargeNumber1" access="VIRTUAL" attribute-type="String">
    <column name="CHARGENUMBER1"/>
    </basic>
    . . .
    </attributes>
  </entity>
  <entity class="com.splwg.oms.model.entity.swman.SwmanSheetView">
    <attributes>
    <basic name="Description" access="VIRTUAL" attribute-type="String">

    <column name="DESCRIPTION"/>
    </basic>
    <basic name="FEEDER_NAME" access="VIRTUAL" attribute-type="String">

    <column name="FEEDER_NAME"/>
    </basic>
    <basic name="SubStation" access="VIRTUAL" attribute-type="String">
    <column name="SUBSTATION"/>
    </basic>
    </attributes>
  </entity>
  . . .
</entity-mappings>
```

SwmanSheet will include all the fields not defined in the code class, but yet are referenced by the JBot sheet configuration.

SwmanStep will include all the fields not defined in the code class, but yet are referenced by the JBot switching step configuration.

SwmanStepView should include the subset of SwmanStep attributes that are populated from model attributes in the ATT_SUPPLEMENTAL_ATTRIBUTES database table or view, any columns displayed in the Misc Log or copied from the Misc Log to a standard sheet, and columns displayed in the Device to Sheet Operation List dialog, which is initiated from the Control Tool's **Switching Plan(s)...** button.

SwmanSheetView should include the subset of SwmanSheet attributes that are displayed in the Open Sheet and New Sheet dialogs.

You need SWMAN_STEP columns for each extra field configured for your condition classes. The columns follow the convention of CONDITION_<field name>, and the SwmanStep class holds them as Condition.<field name>.

5. Add your project specific mappings to the eclipselink-orm.xml file and save it. Double check your mappings to be sure there are no duplicates due to capitalization or inconsistent data. Also check for truncating due to the Oracle 30-character column name limit.

Web Switching/Web Safety Crew Columns

If your project wishes to add additional contractor crew fields, then you will need to be updated following entities using the previously described steps.

```
com.splwg.oms.model.entity.swman.SwmanContractorCrew
com.splwg.oms.model.entity.swman.Crew
```

The swman.SwmanContractorCrew entity maps to the database table SWMAN_CONTRACTOR_CREWS. If you make a change to this entity, then that same change will need to be made to the swman.Crew entity as well.

The swman.Crew entity maps to the database view SWMAN_CREWS_VIEW.

After extending the crew entities you can then reference the new columns anywhere the Web Switching and Web Safety crews are displayed. These entities are only used by Web Switching and Web Safety.

The Product `eclipselink-orm.xml` file has the new entity classes defined and also includes a description of where to add any new columns for these entities.

Global Web Switching Parameters

The following global Web Switching Management rules apply to all sheet types:

SwmanParameters.properties

Rule Name	Valid Values	Description
AutoTransitionSCADASteps	true/false	This parameter defines whether instructed actions sent to SCADA should automatically complete the initiating switching step when the SCADA action is complete. Product has this value set to true.
BroadcastIntervalForUpdates	Number Value	This parameter defines how long (in seconds) the server will gather sheet and safety document updates before broadcasting those updates to the clients. The longer the time period, the better your performance will be, but also the longer the users may have to wait for updates to occur within the Switching Sheet List and Safety Document List. The time specified here is the maximum amount of time the users would have to wait for an update. By default, this parameter will be set to 15 seconds. If a value of 0 is specified, then no broadcasts will occur and the lists will rely on manual refreshes by users. Unless disabled, the minimum value is 5 (seconds); therefore, if the number value is less than 5, it will be converted to 5.
BroadcastSafetyListeners	String Value	A list of tools or adapters that should get safety document updates. If this parameter is not specified, then "SafetyList" is used by default. If your project has an adapter that wants to listen for safety document updates, then that adapter's name needs to be added to this list delimited by a "+" sign. Example: SafetyList+MySafetyAdapter+MyOtherSafety Adapter See the Web Switching and Web Safety Notifications chapter in the *Oracle Utilities Network Management System Adapters Guide* for more details.

Rule Name	Valid Values	Description
BroadcastSwitchingListeners	String Value	A list of tools or adapters that should get switching sheet updates. If this parameter is not specified, then "SwitchingList" is used by default. If your project has an adapter that wants to listen for switching sheet updates, then that adapter's name needs to be added to this list delimited by a "+" sign. Example: SwitchingList+MyAdapter+MyOtherAdapter See the Web Switching and Web Safety Notifications chapter in the *Oracle Utilities Network Management System Adapters Guide* for more details.
CrewFieldName	String - Crew Field	Allows projects to specify what crew field to display in the various crew related fields found in Web Switching and Web Safety. The Crew field to use when populating the switching step, switching audit logs, safety audit logs and event log step entries pertaining to crews. Possible valid values include: `contact`, `crewName`, `crewType`, `comments`, `custom1`, and `custom2`. Only one of the possible values can be specified. Leaving this property blank will result in the value defaulting to `crewName`.
DisableVersioning	true/false	Set this to true when you want to disable all versioning for the switching sheets. A "force" parameter can be used to force the checking in and incrementing of the sheet's version in the command calls to CheckInSheetVersionCommand and IncrementVersionCommand in required places if the global versioning is disabled. By default this option is set to false.
LOADED_SHEETS_LIMIT	Number Value	This is the maximum number of sheets the user can have loaded in NMS at any one time. If the user attempts to load more sheets than this, then they will be issued an error indicating the limit.
OnlyIncludeMiscLogStepsForEventSteps	true/false	When adding steps to the Event Details Steps tab, only pull steps from the Misc Log. If the steps are recorded under another Event's Steps list or are part of another sheet, then leave those steps where they are. The Event Log for the event will still show the operational steps related to that event. When set to false, we'll gather steps from sheets and the Misc Log. We will take any step that has a JobStep relationship to the event in question. By default this option is set to true.
RESET_SAFETY_STATE	Number Value	State key to reset safety documents to during a Training Scenario Reset Model operation.

Rule Name	Valid Values	Description
RESET_SHEET_STATE	Number Value	State key to reset switching sheets to during a Training Scenario Reset Model operation.
SAFETY_ATTRIBUTE_UPDATE. number_of_rules	Number	The number of SAFETY_ATTRIBUTE_UPDATE rules defined. The SAFETY_ATTRIBUTE_UPDATE rules define when device attribute driven fields on a safety document should be updated.
SAFETY_ATTRIBUTE_UPDATE.#.required_doc_states	Comma Delimited List of Safety Document State Keys	One of the states the safety document must be in for the rule to apply. If blank, then all states.
SAFETY_ATTRIBUTE_UPDATE.#.required_doc_types	Comma Delimited List of Safety Document types	The type of safety document this rule should apply to. If blank, then all types.
SAFETY_ATTRIBUTE_UPDATE.#.source_device_handles	Comma Delimited List of Field Names	The device class and index fields from the parent switching sheet.
SAFETY_ATTRIBUTE_UPDATE.#.source_attribute_fields	Comma Delimited List of Field Names	The attribute fields from the database view ATT_SUPPLIMENTAL_ATTRIBUTES. To exclude attributes from the parent switching sheet and only include step (device) column values, then specify "IGNORE". The source_attribute_fields, source_step_fields and target_fields config parameters must have the same number of comma delimited values. All rule parameters must be specified even if you do not intend to assign it a value.
SAFETY_ATTRIBUTE_UPDATE.#.source_step_fields	Comma Delimited List of Field Names	The step field names as defined for each column in the SafetyBody.xml file. To exclude step (device) values, indicate "IGNORE".
SAFETY_ATTRIBUTE_UPDATE.#.target_fields	Comma Delimited List of Field Names	The target fields on the body of the safety document that should be populated with the accumulated data.
SAFETY_ATTRIBUTE_UPDATE.#.only_populate_target_when_blank	true/false	Only populate the target_fields if they are blank.
safety.copy.num_types	Number	The number of safety copy-clear field rules defined. The safety.copy rules define the fields that should be cleared in a safety document when it is copied.
safety.copy.type#.clear_extns	Comma Delimited List	A comma delimited list of safety document extension fields that should be cleared when a safety document is copied.
safety.copy.type#.name	String	The safety document type that has safety copy-clear field rules.

Rule Name	Valid Values	Description
safety.copy.type#.stand_alone_keep_actions	Comma Delimited List	The list of control actions that should be kept when a safety document is copied as a stand alone safety document. For instance, safety documents have Place condition, Remove condition and safety state transition actions, but when copied only the Place condition actions are needed in the new stand alone safety document. Device operation (Open/ Close) actions are prohibited. The format of the values are *<act_cls>:<act_idx>*. act_cls and act_idx pairs are a subset of the actions configured in the swman_safety_type_actions table. Example for a Hold Document: `CONDADD:hold, DDS:EARTH`
Safety.device.status_icon.<EDIT STATE>	String Value - Icon name	This parameter defines the icon name for each of the device list edit states. The device list edit states include: • ADD - The device has been added as part of a viewer device selection. • ADD_STEP - The device has been added as part of a switching sheet step association. • COND_APPLIED - The condition has been applied and updated to the device in the device list. • INCOMPLETE - The device is associated to a switching step where the condition has already been applied to the device. • REMOVE - The device has been marked for removal and will be removed the next time the document transitions from the Unissued to Issued state. • REMOVED - The device has been removed from the device list. These devices are filtered out of the device list.
safety.extensionIgnoreList	Comma Delimited List	List of safety document extension values that may be used internally, but should not be saved to the database. As such, if an extension is in this list and not configured as an extension value in the eclipselink-orm.xml file, then no error will be issued for a missing extension that does not have a matching SWMAN_SAFETY_DOCS database tab
safety.num_types	Number Value	The number of safety document types as defined by the number of safety.type#.name parameters.

Rule Name	Valid Values	Description
safety.type#.link_allowed_states	Comma Delimited List	The comma delimited list of safety document states. If a switching sheet's associated safety document is of the specified type (safety.type#.name) and it's in one of the configured states specified by this parameter, then that safety document will be included in a switching sheet step's Link to Safety Document right click context menu option list. For more details, see description for the Link to Safety Document option described in the *Oracle Utilities Network Management System User's Guide*.
safety.type#.name	String Value	The safety document type. The type# parameter will be "type1" to "typeN" where "N" is the number specified in parameter safety.num_types. Each numbered parameter value will be a single document type. For instance, values could be, but not limited to, HOLD, HOT, CLEAR, INFO. or WARN.
safety.unlinkConditionsFromSheetSteps	true/false	This configuration option defines how a project utilizes conditions and how they relate to switching steps and safety documents. When this option is set to false, specific conditions are tracked in the safety documents and in the switching sheet steps. When the option is set to true, the only thing switching sheet steps and safety documents are looking for is a condition type and not a specific condition. For further details see the documentation for the JBot command StateTransitionCommand. In Product configuration, search for status flag UNLINK_CONDS_FROM_SHEET_STEPS and where it its true value is checked for an example on how to use the parameter. Product uses this JBot status flag to track the setting of this unlinkConditionsFromSheetSteps parameter and configure Product to work appropriately based on the parameter value.
SaveSheetUserPrefs	true/false	Switching sheet preferences are stored for the one sheet that has focus when preferences are stored. Some projects may have significant configuration differences between their sheet types and this can cause issues if a user has a sheet up when their preferences are saved. For these projects, the only option is to not have a sheet up when preferences are saved or turn this option to false to keep your users from accidentally saving switching sheet related preferences that may not apply to all of the project's sheet types. By default, if this property is not set, it will be set to true.

Rule Name	Valid Values	Description
SHEET_ATTRIBUTE_UPDATE.number_of_rules	Number	The number of SHEET_ATTRIBUTE_UPDATE rules defined. The SHEET_ATTRIBUTE_UPDATE rules define when device attribute driven fields on a switching sheet should be updated.
SHEET_ATTRIBUTE_UPDATE.#.required_sheet_states	Comma Delimited List of Sheet State Keys	One of the states the switching sheet must be in for the rule to apply. If blank, then all states.
SHEET_ATTRIBUTE_UPDATE.#.required_sheet_types	Comma Delimited List of Sheet types	The type of switching sheet this rule should apply to. If blank, then all types.
SHEET_ATTRIBUTE_UPDATE.#.source_device_handles	Comma Delimited List of Field Names	The device class and index fields from the switching sheet.
SHEET_ATTRIBUTE_UPDATE.#.source_attribute_fields	Comma Delimited List of Field Names	The attribute fields from the database view ATT_SUPPLIMENTAL_ATTRIBUTES. To exclude attributes from the switching sheet and only include step column values, then specify "IGNORE". The source_attribute_fields, source_step_fields and target_fields config parameters must have the same number of comma delimited values. All rule parameters must be specified even if you do not intend to assign it a value.
SHEET_ATTRIBUTE_UPDATE.#.source_step_fields	Comma Delimited List of Field Names	The step field names as defined for each column in the SwmanSteps.xml file. To exclude step values, indicate "IGNORE".
SHEET_ATTRIBUTE_UPDATE.#.target_fields	Comma Delimited List of Field Names	The target fields on the body of the switching sheet that should be populated with the accumulated data.
SHEET_ATTRIBUTE_UPDATE.#.only_populate_target_when_blank	true/false	Only populate the target_fields if they are blank.
SHEET_STATE_FROM_STEP.#.disallowed_step_states	Comma Delimited List of Step State Keys	This is the list of states that will keep this rule from being applied. If any step is found with any of these states, then the sheet state action will not be applied.
SHEET_STATE_FROM_STEP.#.required_sheet_states	Comma Delimited List of Sheet State Keys	This is the list of sheet states that will allow this rule to be applied. This rule is normally used when there are no steps in the sheet and the state of the sheet has to be used instead. For instance, when removing steps from a sheet.
SHEET_STATE_FROM_STEP.#.required_sheet_type	Comma Delimited List of Sheet types	This is the list of sheet types that will allow this rule to be applied. This is normally used to restrict rule to specific sheet types.
SHEET_STATE_FROM_STEP.#.required_step_states	Comma Delimited List of Step State Keys	This is the list of states that will trigger this rule to be applied. If any step is found with any of these states, then the sheet state action indicated in this rule will be applied to the sheet.

Rule Name	Valid Values	Description
SHEET_STATE_FROM_STEP.#.sheet_state_action	Single Sheet State Action Name	This is the sheet state action name that should be applied to the sheet when the disallowed_step_states and required_step_states checks have passed. See database table TE_STATE_ACTIONS for a list of available actions configured where the APP field is equal to **WSW**. If needed, multiple sheet states to transition through can be provided by listing the sheet states in a comma delimited string. The sheet will be transitioned through the states, from left to right, as they are listed in the string. This may be needed for situations where the plan needs to transition first to an intermediate state before it can be transitioned to the desired final state. An example within product configuration would be to transition to "In-Progress (Instructed)" to "In-Progress(Completed)" to "Completed" after all steps in a plan have been completed successfully.
SHEET_STATE_FROM_STEP.number_of_rules	Number Value	The number of sheet state from step rules configured from 1 to the value configured for this parameter.
SHEET_STATE_TRANS.<Sheet Class #>.<Current Sheet State>.<New State Action Name>.AuditLogMsg = {SWMAN_AUDITLOG.STATE_CHANGE.text}	String Value	This is the audit log message to be populated into the SWMAN_AUDIT_LOG database table. This value will be populated into the LOG_COMMENT record field. This value can include a variable from the sheet tool's property file. In which case, the variable will be substituted for the value defined in the tool's property file. That same property value is normally used for state transition messages configured on the client side.
SHEET_STATE_TRANS.<Sheet Class #>.AuditLogType	String Value	This is the audit log type to be populated into the SWMAN_AUDIT_LOG database table. This value would be populated into the AUDIT_LOG_ENTRY_TYPE record field. For state transitions, this would normally be set to **STATE**.
SHEET_CURRENT_FEEDER_UPDATE.number_of_rules	Number Value	The number of sheet feeder update rules configured from 1 to the value configured for this parameter. The rules are used to populate a switching sheet's Current Feeder and Current Substation fields with values as determined from the sheet's master device and devices used to populate the Steps list. All rule parameters must be specified even if you do not intend to assign it a value. The first rule that matches a sheet's type and states will be used and no further rules will be applied for that sheet.

Rule Name	Valid Values	Description
SHEET_CURRENT_FEEDER_UPDATE.#.required_sheet_types	Comma Delimited List of Sheet Class Numbers	The types of sheets that this rule will apply to. If blank, then this check applies to all sheet types.
SHEET_CURRENT_FEEDER_UPDATE.#.required_sheet_states	Comma Delimited List of Sheet State Keys	One of the listed states the sheet has to be in for the rule to apply. If blank, then this check applies to all sheet states.
SHEET_CURRENT_FEEDER_UPDATE.#.target_feeder_field	String Value	The target field that should be populated with the list of current feeders.
SHEET_CURRENT_FEEDER_UPDATE.#.target_substation_field	String Value	The target field that should be populated with the list of current substations.
SAFETY_CURRENT_FEEDER_UPDATE.number_of_rules	Number Value	The number of safety document feeder update rules configured from 1 to the value configured for this parameter. The rules are used to populate a safety document's Current Feeder and Current Substation fields with values as determined from the document's master device and devices used to populate the Device lists. The document's master device is only used when dealing with stand-alone safety documents. These rules also do not apply if the parent sheet type does not have a rule that will update similar fields. In other words, the parent sheet has to be updated before the associated safety documents can be updated. All rule parameters must be specified even if you do not intend to assign it a value. The first rule that matches a document's type and states will be used and no further rules will be applied for that safety document.
SAFETY_CURRENT_FEEDER_UPDATE.#.required_doc_types	Comma Delimited List of Safety Document Class Numbers	The types of safety documents that this rule will apply to. If blank, then this check applies to all safety document types.
SAFETY_CURRENT_FEEDER_UPDATE.#.required_doc_states	Comma Delimited List of Safety Document State Keys	One of the listed states the safety document has to be in for the rule to apply. If blank, then this check applies to all safety document states.
SAFETY_CURRENT_FEEDER_UPDATE.#.target_feeder_field	String Value	The target field that should be populated with the list of current feeders.
SAFETY_CURRENT_FEEDER_UPDATE.#.target_substation_field	String Value	The target field that should be populated with the list of current substations.
sheet.copy.num_types	Number	The number of sheet copy-clear field rules defined. The sheet.copy rules define the fields that should be cleared in a switching sheet when it is copied.
sheet.copy.type#.class	Number	The class of switching sheet that has sheet copy-clear field rules.
sheet.copy.type#.clear_extns	Comma Delimited List	A comma delimited list of switching sheet extension field names that should be cleared when a switching sheet is copied.

Rule Name	Valid Values	Description
sheet.copy.type#.clear_fields	Comma Delimited List	A comma delimited list of core switching sheet field names that should be cleared when a switching sheet is copied. For example: `completedDate`.
sheet.copy.type#.clear_step_extns	Comma Delimited List	A comma delimited list of switching sheet step extension fields that should be cleared when a switching sheet is copied.
sheet.copy.type#.copy_tables	Comma Delimited List	A comma delimited list of generic tables that should be copied when a switching sheet is copied. See Generic Tables on page 22-35 for information.
sheet.extensionIgnoreList	Comma Delimited List	List of sheet extension values that may be used internally, but should not be saved to the database. As such, if an extension is in this list and not configured as an extension value in the eclipselink-orm.xml file, then no error will be issued for a missing extension that does not have a matching SWMAN_SHEET database table column.
sheet.replication.limit	Number	Replication Switching Sheet Rule. A hard limit on how many switching sheets are allowed to be created based on a replication pattern and date period specified on the Request tab. This is only valid for Planned sheets.
sheet.requireFuzzyAuthority	true/false	If this parameter is set to true, then the user is required to take authority of the FUZZY zone to see switching sheets that are not associated to a modeled device. This parameter only comes into play when the environment is setup to filter switching sheets within the Open Switching Sheet list based on zones of authority.
STAND_ALONE_SAFETY_EDIT_FLAG	JBot Flag Name	This is the flag used to determine if the stand alone safety document has been edited or not. If it has been edited, then change the safety document tab label to an italicized text.
STEP_STATE_COMPLETED	Control Tool Act Key	This act key is used by internal processing to determine if a step has been completed.
STEP_STATE_INSTRUCTED	Control Tool Act Key	This act key is used by internal processing to determine if a step has been instructed.
STEP_STATE_SCADA_INSTRUCTED	Control Tool Act Key	This act key is used by internal processing to determine when a step has been SCADA Instructed.
step.closeActKey	Control Tool Act Key	This act key is assigned to a step when an CLOSE device operation message is processed by the Web Switching service. This can happen when a device is closed by a SCADA system.

Rule Name	Valid Values	Description
step.crew.backToMasterDev	true/false	This option defines whether the crew should migrate back to the switching sheet's master device within the viewer when a step has been instructed and completed. If set to false, the crew will remain on the last instructed device after it has been completed.
step.extensionIgnoreList	Comma Delimited List	List of step extension values that may be used internally, but should not be saved to the database. As such, if an extension is in this list and not configured as an extension value in the eclipselink-orm.xml file, then no error will be issued for a missing extension that does not have a matching SWMAN_STEP database table column.
step.openActKey	Control Tool Act Key	This act key is assigned to a step when an OPEN device operation message is processed by the Web Switching service. This can happen when a device is opened by a SCADA system.
SWMAN_STATES_TO_EXCLUDE_QUERY	String Value	This is the database SQL query to get terminal or final states for sheets which would be excluded while processing model changes.
SWMAN_STEPS_QUERY	String Value	This is the database SQL query to get steps with excluded states and types of sheets. Replace the alias $SWMAN_STATES_TO_EXCLUDE_QUERY$ with a literal list of values for the states to exclude, otherwise the result of SWMAN_STATES_TO_EXCLUDE_QUERY will be used. # Replace the alias $SWMAN_TYPES_TO_EXCLUDE_QUERY$ with a literal list of values for the sheet types to exclude, otherwise the result of SWMAN_TYPES_TO_EXCLUDE_QUERY will be used
SWMAN_TYPES_TO_EXCLUDE_QUERY	String Value	This is the database SQL query to get terminal or final sheet types which would be excluded while processing model changes.
SWMANSHEET_TITLE_JBOT_CONFIG_PARAM	JBot Flag Name	This is the flag to check to determine if the sheet has been edited or not. If it has been edited, then change the sheet tab to an italicized text.

Rule Name	Valid Values	Description
WAIT_TIME_FOR_EVENT_UPDATES	Number Value	The number of milliseconds to wait before submitting a request to gather event and crew information for a switching sheet. During this wait, if another update request comes in then the timer is reset. This feature is in place to eliminate multiple unnecessary requests back to the server for event and crew related updates when the client is receiving a ton of requests. This can happen in a storm or when an event suddenly generates a lot of SRSOutput messages. This timer is not used when first loading a switching sheet. This timer based delay is only used when getting updates dynamically as part of an outside update request. Default value is 1000. A setting of 0 will disable the feature, but will also impact performance of the environment.

Understanding SwmanParameters.properties

For the sheet state transitions (**SHEET_STATE_FROM_STEP**), the **#** values can be a unique sequential number from 1 to the value configured for SHEET_STATE_FROM_STEP.number_of_rules. The rules should be ordered in such a way that when the first rule is determined to be true, then the subsequent rules will be ignored. These rules are used to transition a switching sheet based on the step states. For instance, if a step transitions into the instructed state, then the state of the sheet is set appropriately as well.

> **Note:** Each rule has to include all five of the parameters. Give the parameter a blank value if it is not required, but always include all parameters when specifying a new rule.

Example: setting the sheet state to *In Progress* (Real-Time Instructed Steps) when an instructed step in status 310 is found.

```
SHEET_STATE_FROM_STEP.1.sheet_state_action = instruct_step
SHEET_STATE_FROM_STEP.1.disallowed_step_states =
SHEET_STATE_FROM_STEP.1.required_step_states = 310, 290
SHEET_STATE_FROM_STEP.1.required_sheet_states =
SHEET_STATE_FROM_STEP.1.required_sheet_type =
```

If that rule does not pass, then the next rule is checked until one of the rules causes the sheet state to be updated. If no rule is applied, then no change will be made to the sheet's state.

Example: the sheet is transitioned to the *In Progress* (Real-Time Completed Steps) when there are no instructed steps and at least one step in the state indicated in the required_step_states parameter.

```
SHEET_STATE_FROM_STEP.2.sheet_state_action = complete_step
SHEET_STATE_FROM_STEP.2.disallowed_step_states = 310, 290
SHEET_STATE_FROM_STEP.2.required_step_states = 265, 270, 280, 320, 365
SHEET_STATE_FROM_STEP.2.required_sheet_states =
SHEET_STATE_FROM_STEP.2.required_sheet_type =
```

Example: if the sheet is in the *In Progress* (Real-Time Instructed Steps) state and you don't have any instructed steps, then we transition to the *In Progress* (Real-Time Completed Steps) state.

```
SHEET_STATE_FROM_STEP.3.sheet_state_action = complete_step
SHEET_STATE_FROM_STEP.3.disallowed_step_states = 310
SHEET_STATE_FROM_STEP.3.required_step_states =
SHEET_STATE_FROM_STEP.3.required_sheet_states = 210
SHEET_STATE_FROM_STEP.3.required_sheet_type =
```

Example: in this scenario, we describe how to configure the audit log messages for the Emergency switching sheet when it goes into the *In Progress* state.

```
# Status 50 - New (Emergency)
# Status 220 - Hold/Pending
SHEET_STATE_TRANS.3109.AuditLogType = STATE
SHEET_STATE_TRANS.3109.50.instruct_step.AuditLogMsg =
{SWMAN_AUDITLOG.STATE_CHANGE.text}
SHEET_STATE_TRANS.3109.50.complete_step.AuditLogMsg =
{SWMAN_AUDITLOG.STATE_CHANGE.text}
# emerg_active is used when creating an Emergency sheet from an event.
SHEET_STATE_TRANS.3109.50.emerg_activate.AuditLogMsg =
{SWMAN_AUDITLOG.STATE_CHANGE.text}
SHEET_STATE_TRANS.3109.220.instruct_step.AuditLogMsg =
{SWMAN_AUDITLOG.STATE_CHANGE.text}
SHEET_STATE_TRANS.3109.220.complete_step.AuditLogMsg =
{SWMAN_AUDITLOG.STATE_CHANGE.text}
```

The Product Emergency switching sheet uses class number 3109. Specifying the current state of the sheet as part of the parameter name, is optional. If the current state is not given, then any time the sheet action instruct_step, emerg_activate, or complete_step are initiated, the audit log will be created. Since a sheet can jump between the states for the instruct_step and complete_step, we need to specify the from state value. Otherwise we would get a new audit log message each time a step in the sheet is instructed and completed.

Since we use variables for AuditLogMsg, the actual audit log message has to be looked up in the SwmanEmergencyTool_en_US.properties file. For Product this values is set to

```
SWMAN_AUDITLOG.STATE_CHANGE.text = Switching Sheet State Change
```

At this time, comments from other fields on the switching sheet cannot be incorporated into the server based audit log messages. That type of audit log message still has to be initiated from the client side through a JBot command called AddAuditLogDetailsCommand.

Note Concerning Control Action Descriptions

In order to have switching steps added to the Event Log and a new switching sheet created from an event (through Event Details or the Work Agenda), the control action records associated to those steps have to have "Description" entries configured. Within the CONTROL_ACT table, the control action records have to have a "description" value.

For example, if a SCADA operation creates an outage and that operation's control action does not have a description, that operation will not be added to the Event Log. If an operator creates an Emergency switching sheet from the outage in the Work Agenda, the new sheet will not have that initial SCADA operation listed as a step. Conversely, if the operation has a description, the Event Log and Emergency switch sheet will list the SCADA operation.

GUI Configuration Overview

The bulk of the Web Switching and Web Safety GUI configuration can be found in the jconfig/ops/webswitching directory. The configuration is spread across many files. This allows projects to customize bits and pieces of the configuration without having to define a custom version of the entire tool configuration. If at all possible, projects should inherit from Product as much as possible so that upgrades and patch installations are more easily applied to a project. The following tables describe the main modules used to configure the applications. Each of the configuration modules has an xml and properties file associated to it.

Web Switching

Name	Description
SwmanEntities.inc	This file includes a number of XML entities that are used throughout the Web Switching xml configuration files. The entities are used to give state key numbers readable names so that the configuration can be more easily followed. Instead of displaying a number in the configuration, a name is displayed. Changing the states and such to reference one entity also makes updating the configuration easier. If your project has defined a different switching sheet state for instance, then the single entity will only need to be altered instead of each instance where the entity is referenced.
SwmanBaseProperties	This module defines all the imports, datastores and dialogs used by each of the switching sheet types.
SwmanEmergencyTool SwmanPlannedTool SwmanMiscLogTool SwmanTemplateTool SwmanOutageCorrectionTool SwmanTrainingTool SwmanVoltVarTool SwmanLoadShedTool	Each of these modules defines the tool behavior for each of the switching sheet types. The modules also include all of the other modules used to build the GUI configuration for each of the switching sheet types.
SwmanToolBar	The switching sheet Menu/Toolbar configuration.
SwmanRequest	The switching sheet's Request tab configuration.
SwmanSteps	The switching steps Table configuration. Each of the step columns are defined in this module.
SwmanStepsPopupMenu	The right click switching Steps table context menu configuration.
SwmanStepsHeader	The switching steps toolbar button definitions.
SwmanHeader	The switching steps Event List and Crew List configuration.
SwmanEventsPopupMenu	The right click Events List table context menu configuration.
SwmanCrewsPopupMenu	The right click Crew List table context menu configuration.

Name	Description
SwmanImpactedCustomers	The switching sheet's Impacted Customers tab configuration.
SwmanImpactedCustomersPopupMenu	The right click Impacted Customers table context menu configuration.
SwmanActions	Add your reusable <Action> elements to this file. Also include any changes to actions used by the code itself, like the ACT_APPLY_SAFETY_FILTERS.
SwmanSheetOverlaps	The switching sheet's Overlaps tab configuration.
SwmanExternalDocuments	The switching sheet's External Documents tab configuration.
SwmanExternalDocumentsPopupMenu	The right click External Documents table context menu configuration.
SwmanViewAreas	The switching sheet's View Areas tab configuration.
SwmanViewAreaPopupMenu	The right click View Areas table context menu configuration.
SwmanSafety	The switching sheet's Safety Documents tab configuration.
SwmanTracking	The switching sheet's tracking panel configuration on the Tracking/Audit Log tab.
SwmanAuditLog	The switching sheet's audit log panel configuration on the Tracking/Audit Log tab.
SwmanStatusBar	The switching sheet's status bar configuration.

Web Safety

Name	Description
SafetyBaseProperties	This module defines all the imports, datastores and dialogs used by each of the safety document types.
SafetyStateTransActions.inc	This file includes a set of actions pertaining to safety state transitions. The action names follow the format of *SAFETY_STATE_ACTION_<New State>.* These action names are looked up by the code, so the format of the action name has to be as specified. The actions each contain a list of commands that will be initiated when a safety document transitions to a new state. Each of the actions take one parameter called *$FROM_SWITCHING$*, which indicates whether the request is coming from Web Switching Management or not. See document for command `StateTransitionCommand` for more information. The file also includes an action defining the `actKey` to use when recording devices into the Tag Points device lists through view selections. This `actKey` pertains to the control action used to apply the appropriate tag to the device when the document is issued.
SafetyStateTransValidationRules.inc	This module defines the validation rules for the group name `CHECK_FOR_DEVICES_AND_CREWS`. This validation group is normally initiated when a safety document transitions to a new state. This can happen when executing steps from a sheet or from a safety document.
SafetyTool	This module defines the tool behavior for each of the safety document types. The modules also include all of the other modules used to build the GUI configuration for each of the safety document types.
SafetyTitle	The title configuration for all the safety document types.
SafetyToolbar	The safety document toolbar configuration.
SafetyBody	The document configuration for each of the safety document types. This module includes conditional checks for each of the safety document types to determine when to display components on the GUI as not all of the safety documents have the exact same GUI layout.

Name	Description
SafetyDeviceListPopupMenu	This is the popup menu in the Tag Points device list. It includes actions to remove, view and model verify the device.
SafetyGroundsListPopupMenu	This is the popup menu in the Grounds device list. It includes actions to remove, undo, view and model verify the device.
SwmanStandAloneSafetyTool	Defines the tool behavior for the stand alone safety documents. This module also includes all of the other modules used to build the GUI configuration for this tool.
SwmanStandAloneSafetyMenuToolBar	Defines the MenuToolbar for the stand alone safety documents. This includes items like the Save and Exit options.
SwmanStandAloneSafetyPopupMenu	This is the popup menu in the Step Actions list. It currently only has one option and that's to verify individual steps when they have been impacted by a model edit.
SwmanStandAloneSafetyProperties	This module defines all the imports, datastores and dialogs used by the stand alone safety documents.
SwmanStandAloneSafetyStatusBar	The stand alone safety document's status bar configuration.
SwmanStandAloneSafetyStepsToolbar	The switching steps toolbar button definitions for the associated steps list.
SwmanStandAloneSafetyToolbar	The stand alone safety document toolbar configuration.

Switching Sheets

Automated Emails

Web Switching along with BI Publisher reports, supports the ability to automatically send emails triggered by the user through button selections or through state transitions of a switching sheet. The emails can be setup to be sent to a pre-configured recipient and/or addresses pulled from the switching sheet's data fields.

It should be noted that these automated email requests are fulfilled from WebLogic and the user's email client is not involved in this method of emailing switching sheets.

If any value specified in the TO, CC or BCC fields of the email are not properly formatted like an email address, then this will result in the email not being sent. If the email request was initiated from the NMS client application,then an error dialog will be displayed to the user. If the emails are part of a back-end state transition request, then an error will be displayed in the server log.

One method for initiating the automated emails is to configure the action as part of the sheet's TE state transitions. Below is an example of the callback and arguments used to generate the automated email behind the scenes. Any errors will not be displayed back to the user.

```
INSERT INTO te_state_callbacks
  (app, cb_key, state_key, condition, action, abort_on_fail, can_undo,
   error_code)
VALUES
  ('WSW', 155, 170, 'PRE_ENTER', 'email', 'Y', 'N', -155);

INSERT INTO te_state_cb_args
  (app, cb_key, arg_key, arg_name, arg_value)
VALUES
  ('WSW', 155, 140, 'REPORT_TYPE', 'SwitchSheetsStateEmail');

INSERT INTO te_state_cb_args
  (app, cb_key, arg_key, arg_name, arg_value)
VALUES
  ('WSW', 155, 150, 'REPORT_SUBTYPE',
'$swmanSheetCls.switchSheetCls$');
```

Another method is to use the JBot command ExecuteStateCallbackCommand. In this case you'll need to make a call to the callback method SendEmailJob and pass in arguments like what was done with the TE state arguments. This JBot command allows you to use the same configuration to initiate state transition based emails along with client initiated emails based on user actions.

```
<Command value="ExecuteStateCallbackCommand">
    <Config name="method_name" value="SendEmailJob"/>
    <Config name="arg_names" value="REPORT_TYPE, REPORT_SUBTYPE"/>
    <Config name="arg_values" value="SwitchSheetsStateEmail,
$swmanSheetCls.switchSheetCls$"/>
</Command>8
```

Once the initiating calls are in place, the back-end configuration needs to be applied. This all occurs from within the SwmanUserText_en_US.properties file. Below is an example of the different options and arguments used to create the emails.

```
# Automatic emailing of switching report. The email will only be
```

```
# sent out if the email_to field value is not empty. Variables are
derived
# from the SwmanSheet object and not a datastore, thus the variables
are
# wrapped in "$" symbols. Extension values have to start with
"extensions.".

# BI Publisher report location, template name and type.
BIP_GENERATE_REPORT.SwitchSheetsStateEmail.3117.report_location = /
WebSwitching/Templates.xdo
BIP_GENERATE_REPORT.SwitchSheetsStateEmail.3117.report_template =
PlannedSheet
BIP_GENERATE_REPORT.SwitchSheetsStateEmail.3117.report_format = pdf

# The optional user_flag is a value on the sheet that the user can
alter
# to disable automatic emails for this request. The flag has to have a
# value of Y, Yes, or True to cancel the email request.
#BIP_GENERATE_REPORT.SwitchSheetsStateEmail.3117.user_flag =
$extensions.disableEmailNotifiations$

# Indicate report parameters here delimited by semicolons.
# Example: SHOW_STEPS=false; SHOW_DEBUG=true;
BIP_GENERATE_REPORT.SwitchSheetsStateEmail.3117.report_params =
SHEET_CLS=$swmanSheetCls.switchSheetCls$; SHEET_IDX=$switchSheetIdx$;
ATTACH_EXTERNAL_DOCUMENTS=true;

# The filename of the report to generate.
BIP_GENERATE_REPORT.SwitchSheetsStateEmail.3117.base_file_name =
$swmanSheetCls.switchSheetType$_$switchSheetIdx$

# The email subject.
BIP_GENERATE_REPORT.SwitchSheetsStateEmail.3117.email_subject =
Scheduled Switching Sheet - $swmanSheetCls.switchSheetType$
$switchSheetIdx$


# If an email_from value is not given, then the user's login ID is used
along with
# the username_to_email database query specified in the
CentricityServer.properties file.
# That query can be used to apply a domain to the user's ID or to
translate a user's ID
# to a valid email address. If your project prefers to keep the "from"
email address
# generic, then uncomment the following parameter and specify an email
address.
#BIP_GENERATE_REPORT.SwitchSheetsStateEmail.3117.email_from =
no_reply@somedomain.com
BIP_GENERATE_REPORT.SwitchSheetsStateEmail.3117.email_to =
$extensions.RequesterEmailAddress$
#BIP_GENERATE_REPORT.SwitchSheetsStateEmail.3117.email_cc =
$extensions.EmailAddress2$
#BIP_GENERATE_REPORT.SwitchSheetsStateEmail.3117.email_bcc =
$extensions.GroupEmailAlias$

# This is text that will show up in the body of the email that will be
sent
# out. The Switching Sheet report will be an attachment to the email.
If
# external documents are included, then they will be attached to the
email
```

```
# as well.
BIP_GENERATE_REPORT.SwitchSheetsStateEmail.3117.email_body = Email
body message for $swmanSheetCls.switchSheetType$
$switchSheetIdx$.\n\nSwitching Sheet Description:
$extensions.Description$.
```

Sheet Types

Each of the switching sheet types are defined in the SWMAN_SHEET_CLS configuration table. Each switching sheet type has its own class. See Configuring Classes and Inheritance for further details on adding a class.

Within the SWMAN_SHEET_CLS configuration table, define which JBot tool configuration should be used when the sheet is loaded within the Web Workspace environment. The switching sheet types can either share the same configuration or have their own. For instance, multiple Planned switching sheet types can all use the same SwmanPlannedTool definition and then within the tool configuration, define minor differences between the types based on the class of switching sheet being displayed.

State Transitions

State transitions for the switching sheets and their individual steps are all configured in the TE State Transition database tables where the *app* value to each of the tables is set to *WSW*.

> **Note:** See tables te_valid_states, te_status_groups, te_statuses, te_state_transitions, te_state_actions, te_expressions, te_init_state_rules, te_state_callbacks, and te_state_cb_args for more information.

Do not cross reference step and sheet states. Keep them completely separate. For example, create a state for the step Completed state and another state for the sheet Completed state. Do not try to use a single state for both the sheets and the steps.

Web Switching sheets support the following callbacks.

Callback Action Name	Description
safety_state_check	Determine if the sheet's associated safety documents are in the completed state. Switching sheets should not be completed when there are outstanding safety documents still issued to crews.
unrestored_pln_check	Determine if the switching sheet has any unrestored Planned events associated to it. In most cases, Planned switching sheets should not leave customers out of power.
create_switching_job	Create the Master switching sheet event that is normally used for Planned switching sheets.
complete_switching_job	Complete the Master and any Planned events associated to the switching sheet. This callback is normally used by Planned switching sheets.
remove_crews	Removes the crews that are associated to a switching sheet. This is normally initiated after a switching sheet has been completed.
update_dms_job	Update a dms job associated with the switching sheet. This is used by VVO.

The following is an example for the Issue state:

```
INSERT INTO te_state_callbacks
  (app, cb_key, state_key, condition, action, abort_on_fail, can_undo,
   error_code)
VALUES
  ('WSW', 130, 232, 'PRE_ENTER', 'safety_state_check', 'Y', 'N', -
130);
INSERT INTO te_state_callbacks
  (app, cb_key, state_key, condition, action, abort_on_fail, can_undo,
   error_code)
VALUES
  ('WSW', 140, 232, 'PRE_ENTER', 'unrestored_pln_check', 'Y', 'N', -
140);
INSERT INTO te_state_callbacks
  (app, cb_key, state_key, condition, action, abort_on_fail, can_undo,
   error_code)
VALUES
  ('WSW', 160, 232, 'PRE_ENTER', 'complete_switching_job', 'Y', 'N', -
160);
```

The error_codes are used to display distinct dialog messages to the user when the action fails. The messages for these error codes are configured in the MessageCode_en_US.properties file.

The following is an example for error code "-130", which was referenced in the above te_state_callbacks example.

```
OmsClientException.WSW.STATE.CALLBACK.130 = Not all safety documents
are completed
OmsClientException.WSW.STATE.CALLBACK.130.title = State Transition
Failed
```

Sheet Data Fields

Data fields in this case are in reference to the fields found on the Request tab of the sheet. Data fields can be found anywhere on the sheet, but Product configuration has grouped the majority of them to one tab. The data fields are stored in the SWMAN_SHEET table.

The following is an example of how to reference a value from the table.

SWMAN_SHEET

For core values not defined in the eclipselink-orm.xml file, the following format should be used.

```
data_source="DS_SWITCHING_SHEET_LOCAL.deviceAlias"
```

For any fields defined in the eclipselink-orm.xml file, use the following format:

```
data_source="DS_SWITCHING_SHEET_LOCAL.extension.RequesterPhoneNumber"
```

For more information on the list of available data source values, refer to the DS_SWITCHING_SHEET datastore documentation.

Open Switching Sheet List

The Open Switching Sheet list is populated through the DS_OPEN_SWITCHING_SHEET_LIST datastore, which is populated from the view SWMAN_SHEETS_LIST. The amount of data displayed in this list should be kept to a reasonable level. The more data that is displayed, the longer it will take the dialog to be displayed. The list of sheets populated into the list should be limited as much as possible for performance reasons. This is done by specifying a **where** clause in the "additional_constraints" configuration parameter of the DisplayOpenNMSDialogCommand command. This **where** clause is applied when querying the SWMAN_SHEETS_LIST database view. Projects can configure any number of calls to this command with unique **where** clauses and have separately configured options on the Open Switching Sheet list to display different sets of switching sheets.

The Open Switching Sheet list is initiated from the Web Workspace **File** menu or toolbar. The command that initiates that request is DisplayOpenNMSDialogCommand. Refer to the NMS Commands documentation for further details about this command.

Not only should the **where** clauses be used to limit the amount of data being passed to the client, but the database view SWMAN_SHEETS_LIST should also be defined with only the extension fields that are displayed on the Open Switching Sheet list. Query for data not displayed on the GUI is wasteful and should be avoided.

New Switching Sheet List

The New Switching Sheet type list is populated through the DS_SWITCHING_SHEET_CLS datastore, which is populated from the SWMAN_SHEET_CLS database table. The pre-created sheet list displayed on this dialog is populated through the DS_NEW_SWITCHING_SHEET_TEMPLATE_LIST datastore, which is populated from the database view SWMAN_SHEETS_LIST. The amount of data displayed in this list should be kept to a reasonable level. The more data that is displayed, the longer it will take the dialog to be displayed.

The New Switching Sheet list is initiated from the Web Workspace **File** menu or toolbar. The command that initiates that request is DisplayNewNMSDialogCommand. This command accepts a **where** clause to use to gather the data from the database. The **where** clause is configured with the command's "additional_contraints" configuration parameter. The same **where** clauses used by the DisplayOpenNMSDialogCommand can be used with this command as well. Refer to the NMS Commands documentation for further details about this command.

Search Switching Sheet

The Search Switching Sheet list is populated through the DS_SEARCH_SWITCHING_SHEET_LIST, which is populated from the SWMAN_SHEETS_LIST database view. The amount of data displayed in this list will be limited to 1000 entries.

The GUI and command calls are configured in the DLG_SEARCH_SWITCHING_DIALOG.xml file.

The Search Switching Sheet list is initiated by selecting **Switching Sheet Search...** from the **Open** option in the Web Workspace **File** menu. The SearchSwitchOrSafetyCommand command requests the search results. This command has a parameter called "search_fields" that limits the number of fields to search in within the SWMAN_SHEET_LIST database view. It is strongly advised that your project keep this list to a minimum since the more fields to be searched, the more intensive the query will be on the system.

Device to Sheet Operation List

The Device to Sheet Operation List is launched from the Control Tool for a device selected in the Viewer. The Device to Sheet Operation List is populated through the DS_DEVICE_SHEETS_LOCAL datastore, which is populated from the SWMAN_DEVICE_SHEET database view. The list displays the switching actions associated to the device. The view is configured to gather all the steps associated to active switching sheets and to also pull in the last 30 days worth of completed actions as well. The filtering criteria are defined in the oracle view, which can be redefined by copying the view definition from the `nms_schema_web_swsheets.sql` file and placing it into the project version of this file.The Product dialog DLG_DEVICE_SHEETS has been configured to only pull in the sheet extension value called "Description." If projects require additional fields in this dialog, then they will have to create a custom project version of this dialog and add the necessary fields to the table defined in the dialog.

Model Verification

The Web Switching service initiates a query each time it receives a notification of a model build or edit. When this notification comes through, the following query is initiated:

```
SELECT sheet.switch_sheet_cls, sheet.switch_sheet_idx,
       step.step_cls, step.step_idx
FROM swman_sheet sheet, swman_step step,
     network_components nc, swman_patches sp
WHERE sheet.seq_sheet_id = step.seq_sheet_id AND
      // Exclude Block steps
      step.parent_step_id IS NOT NULL AND
      ( (step.dev_cls = nc.h_cls AND step.dev_idx = nc.h_idx) OR
        (step.gnd_node_cls = nc.port_a_cls AND
         step.gnd_node_idx = nc.port_a_idx) OR
        (step.gnd_node_cls = nc.port_b_cls AND
         step.gnd_node_idx = nc.port_b_idx) ) AND
      (nc.death > sp.patch_time OR  nc.birth > sp.patch_time) AND
      // Where the sheet and step are not in a termination state
      step.state_key NOT IN (<<List of terminal step states>>) AND
      sheet.state_key NOT IN (<<List of terminal sheet states>>) AND
     sheet.switch_sheet_cls not in (<<Outage Correction Sheet Types>>)
ORDER BY step.seq_sheet_id, step.step_idx
```

The MB_EDIT field in the SWMAN_STEP table is updated for each of the step records returned by this query. These steps will have to be validated by the user before switching sheet step executions can continue in the switching sheet.

The switching sheet and step terminal states are determined by running the following query:

```
SELECT state_key FROM te_valid_states tevs
WHERE tevs.app='WSW' AND tevs.state_key NOT IN
    (SELECT DISTINCT from_state_key FROM te_state_transitions te
     WHERE te.app='WSW')
```

Default Crews

To configure default crews for switching sheets, add the following sections to [project]_web_swsheets.sql:

```
DELETE FROM swman_crews WHERE seq_sheet_id = 0;
INSERT INTO swman_crews(seq_sheet_id, crew_key)
SELECT 0, crew_key FROM crews WHERE active='Y' AND
crew_name IN ('Default_Crew_1', 'Default_Crew_2');
```

Replacing '*Default_Crew_1*', '*Default_Crew_2*' with a comma-separated list of the names of the default crews as they exist in the crews table.

The SWMAN_CREW_EVENT database view will also need to be updated to include the crews that were previously inserted into the SWMAN_CREWS table. Do this by including an "or" statement at the end of view definition so that all crews with seq_sheet_id equal to 0 are included.

```
or sc.seq_sheet_id = 0
```

The view definition can be found in file [project]_schema_web_swsheets.sql.

Versioning

Switching sheet versioning can occur manually or automatically. Product configuration is setup to automatically check in the switching sheet when it reaches the Issued state. This is done by initiating a call to the command CheckInSheetVersionCommand.

The version of a switching sheet will be automatically incremented when steps are manipulated (added, cut, pasted or deleted) within the sheet and when the switching sheet field CHECKED_IN has been set to *Y*. This field is stored in the SWMAN_SHEET table. The JBot flag VERSION_CHECKED_IN is set based on the value of the CHECKED_IN field. This flag is used by the JBot configuration to determine when to initiate commands based on version control.

Product configuration has been setup to increment the version automatically if any of the fields on the Request tab are altered. This is done by initiating the call to the command IncrementVersionCommand. This command will only execute if the switching sheet's CHECKED_IN database field is set to *Y*.

The current version of the switching sheet is stored in the REVISION field of the SWMAN_SHEET database table.

Versioning can be disabled globally for all the switching sheets through a global parameter DisableVersioning found in `SwmanParameters.properties` file. By default, Product Configuration has this value set to false. To disable all the versioning, one can set the DisableVersioning value to true.

If there is a requirement to enable versioning for a specific action when the global versioning is disabled(DisableVersioning is set to true), a "force" parameter can be used to force the checking in and incrementing of the sheet's version in the command calls to CheckInSheetVersionCommand and IncrementVersionCommand in required places.

For example:

```
<Command value="CheckInSheetVersionCommand">
    <Config name="force" value="true"/>
</Command>
```

Overlaps

The switching sheet overlaps list uses the DS_OVERLAPS datastore. This datastore is populated from the SWMAN_OVERLAPS database view. The database view is defined in the `product/sql/nms_schema_web_swsheets.sql` file. This same view is used by the Global Overlaps list, so any changes to this view will impact that list as well.

Product is configured to only include sheets classified under the categories of PLANNED and EMERGENCY. The list is also filtered based on the state of the sheet. The list of state keys is included in the view definition. If any switching sheet states have been added to a projects configuration, this view may need to be redefined by the project.

External Documents

The switching sheet external documents list uses the DS_EXTERNAL_DOCUMENTS datastore. This datastore is populated from the SWMAN_SHEET_DOCUMENTS database table.

This functionality can be limited to certain file types. See the Attachment Security on page 3-18 for more details.

The External Documents functionality cannot be altered other than to change the column labels and sensitivity of the button options. The command DisplayFileChooserCommand, is used to gather files to be included in the list. Any changes to the file list are not saved to the database until the switching sheet is saved.

Generate Isolation Steps

The JBot command GenerateIsolateStepsCommand is used from the Control Tool to create a set of steps to isolate a piece of conductor within the model. The steps are generated based on the session the command was initiated from. If the Control Tool is in Real Time, then the Real Time model is used. If the Control Tool is in Study mode, then the user's study session is used. You also need to have a switching plan pre-created and in record mode in order to accept the generated steps. Both the session and the switching sheet requirements cannot be altered.

At this time, the command only supports isolating a conductor. The command uses the classes ss_isolate and ss_secure to determine what device types to create switching steps for. The command arguments determine the types of steps to generate for the isolate and secure device types. For more information, see the command documentation and the description of ss_isolate and ss_secure classes in the Configuring Classes and Inheritance section (see Configuring Classes and Inheritance on page 22-2 for information).

Generic Tables

You can add generic tables to your Switch Sheets by following these steps:

1. Create your retain database table to hold the data. This should include the SEQ_SHEET_ID column, so that these records are tied to the Switch Sheet.
2. Add a DragSource to the source table in that tool. For example, in the DDSAlarms.xml, you can add the following to the table definition:

```
<DragSource flavor="DDS_ALARM" enable_copy="true"
  enable_cut="false"/>
```

3. Create a new JDBCDataStore in your Switch Sheet. For example:

```
<DataStoreClass name="DS_SWMAN_ALARMS" scope="local"
  class="com.splwg.oms.jbot.JDBCDataStore"
                table="SWMAN_SHEET_ALARMS">
   <PrimaryKey value="SEQ_SHEET_ID"/>
   <PrimaryKey value="ALARM_CLS"/>
   <PrimaryKey value="ALARM_IDX"/>
</DataStoreClass>
```

4. Create your destination table in your Switch Sheet. This should include the DragPerform. For example:

```
<Table name="TBL_QA_ALARMS">
   <TablePlacement anchor="NORTHWEST" fill="BOTH" height="1"
    insets="2,2,2,2" ipad="1,1" start="0,1" weight="1,1" width="10"/>
   <TableBehavior auto_resize_columns="true" click_to_sort="true"
    display_header="true" intercell_spacing="1,1"
    reorder_columns="true" resize_columns="true"
    selection_policy="multiple" row_height="16"
    row_selection_allowed="true" show_horizontal_lines="true"
    show_vertical_lines="true" unselect_if_filtered="false"
    even_bg_color="224,224,224" data_source="DS_SWMAN_ALARMS">
      <Column key="SYSTEM_TIME">
      <Editable initial="true"/>
      </Column>
      <Column key="FEEDER">
      <Editable initial="true"/>
      </Column>
      <Column key="ALIAS">
      <Editable initial="true"/>
      </Column>
      <Column key="DESCRIPTION">
      <Editable initial="true"/>
      </Column>
      ...
      <DropPerform flavor="DDS_ALARM">
         <Command value="CopyRowsCommand">
           <Config name="source_datastore" value="DS_DDS_ALARMS"/>
           <Config name="destination_datastore"
            value="DS_SWMAN_ALARMS"/>
           <Config name="columns" value="eventHdl.classNumber,
            eventHdl.instanceNumber,eventType,#DATE_TIME,
            #feeder,alias,description,#source
            #state,phases,#attributeName,#nominal,quality,
            limit,whoAck,timeAck,objectHdl.classNumber,
            objectHdl.instanceNumber,rtiAlias"/>
           <Config name="destination_columns" value="ALARM_CLS,
            ALARM_IDX,ALARM_TYPE,SYSTEM_TIME,FEEDER,
            ALIAS,DESCRIPTION,SOURCE,STATE,PHASES,
```

```
                ATTRIBUTE_NAME,NOMINAL,QUALITY,LIMIT,
                WHO_ACK,TIME_ACK,DEVICE_CLS,
                DEVICE_IDX,RTI_ALIAS"/>
              <Config name="common_sources"
               value="DS_SWITCHING_SHEET_LOCAL.seqSheetId"/>
              <Config name="common_targets" value="SEQ_SHEET_ID"/>
              <Config name="no_selection_msg"
               value="SWITCHING.NO_DDS_ALARMS_SELECTED"/>
              <Config name="duplicate_row_msg"
               value="SWITCHING.DUPLICATE_DDS_ALARMS"/>
            </Command>
        </DropPerform>
        <Perform name="CellEdited" category="cellEdited">
          <Command value="JDBCSaveCommand">
            <Config name="datastore" value="DS_SWMAN_ALARMS"/>
          </Command>
        </Perform>
    </TableBehavior>
</Table>
```

You can also add buttons to copy and remove rows. For example:

```
<Button name="BTN_ADD_ALARM">
    <ButtonPlacement anchor="NORTHWEST" start="0,0"/>
    <ButtonBehavior icon="add.png">
        <Enabled when="PLAN_LOCKED"/>
        <PressPerform>
        <Command value="CopyRowsCommand">
            <Config name="source_datastore" value="DS_DDS_ALARMS"/>
            <Config name="destination_datastore"
             value="DS_SWMAN_ALARMS"/>
            <Config name="columns"
             value="eventHdl.classNumber,eventHdl.instanceNumber,
              eventType,#DATE_TIME,#feeder,alias,description,
             #source,#state,phases,#attributeName,
             #nominal,quality,limit,whoAck,timeAck,
             objectHdl.classNumber,objectHdl.instanceNumber,rtiAlias"/>
            <Config name="destination_columns"
             value="ALARM_CLS,ALARM_IDX,ALARM_TYPE,SYSTEM_TIME,
             FEEDER,ALIAS,DESCRIPTION,
             SOURCE,STATE,PHASES,ATTRIBUTE_NAME,
             NOMINAL,QUALITY,LIMIT,WHO_ACK,TIME_ACK,
             DEVICE_CLS,DEVICE_IDX,RTI_ALIAS"/>
             <Config name="common_sources"
              value="DS_SWITCHING_SHEET_LOCAL.seqSheetId"/>
             <Config name="common_targets" value="SEQ_SHEET_ID"/>
             <Config name="no_selection_msg"
              value="SWITCHING.NO_DDS_ALARMS_SELECTED"/>
             <Config name="duplicate_row_msg"
              value="SWITCHING.DUPLICATE_DDS_ALARMS"/>
        </Command>
        <Command value="JDBCSaveCommand">
            <Config name="datastore" value="DS_SWMAN_ALARMS"/>
        </Command>
        </PressPerform>
    </ButtonBehavior>
</Button>
<Button name="BTN_DELETE_ALARM">
```

Switching Steps

Manual Step Addition

Web Switching normally records steps initiated from the Control Tool. So as not to clutter the Control Tool menu options, steps can also be added to a switching sheet's steps list using the Manual Step Addition (MSA) form. The form is capable of adding actions that may or may not be available from the Control Tool. The list of available actions are also filtered based on the device that is selected. If you have actions configured on the Control Tool and the control action has a switching code associated to it, then that action will be available for selection within the MSA form.

The first step in configuring an action for inclusion on the MSA form is to give the associated control action record in the CONTROL_ACT table a unique switching_code value. In the following example the action Check Open has a switching code equal to 'co'.

```
INSERT INTO control_act
  (act_key, act_cls, act_idx, action_name, label, instruct_label,
   switching_desc, switching_code, description, undo_act_key)
VALUES
  (1550, 'HLMsg', 'NOOP', '', 'Check open', 'Instruct Check open',
   '#swmanStep.deviceAlias%Check open {0}', 'co', 'Check open $ALIAS',
0);
```

Once your action have been configured in the control_act table, the next step is to configure a popup menu item on the Control Tool. The action does not need to be visible to the user from the Control Tool, but must be associated to a popup item or button on that tool. The Control Tool configuration for the button defines when the option is available for selection within the MSA form. For options that should not be visible to the user, then add the items within the HIDDEN_MANUAL_POPUP section within your Control.xml configuration. For instance the Check Open option is not available to the user on the Product Control Tool GUI, but if a switchable device has been populated into the MSA form, then the following operation will be available to the user:

```
        <PopupMenuItem class="javax.swing.JMenuItem"
name="BTN_MANUAL_CHECK_OPEN">
          <Enabled initial="true"/>
          <Visible initial="true"/>
          <ControlActions>
             <ControlAction key="1550"
when="{DS_CONTROL_TOOL.DEVICE_CLASS_PARENTS == 'switch'}"/>
          </ControlActions>
          <PressPerform>
          </PressPerform>
        </PopupMenuItem>
```

The above Menu Item allows control action 1550 (Check Open) to be available to every device that inherits from class 'switch'.

State Transitions

State transitions for the switching sheet steps are all configured in the TE State Transition database tables where the *app* value to each of the tables is set to *WSW*.

> **Note:** See tables te_valid_states, te_status_groups, te_statuses, te_state_transitions, te_state_actions, te_expressions, te_init_state_rules, te_state_callbacks, and te_state_cb_args for more information.

Do not cross reference step and sheet states. Keep them completely separate. For example, create a state for the step Completed state and another state for the sheet Completed state. Do not try to use a single state for both the sheets and the steps.

Control Tool Actions

Web Switching Management uses the same rules that the Control Tool uses to determine if a control action is valid or not. Product configuration is configured to keep the two tools in synch. If the Control Tool does not allow an Open operation on a device, then a switching step with that same action will not allow the operation either. To get around this, the JBot flag FROM_SWITCHING can be used within the control/xml/Control.xml file and its include files to give actions alternate rules when the action originates from Web Switching Management. For more information, see the Control Tool Configuration chapter.

Step Columns

Switching step column data is stored in the SWMAN_STEP table. Here are examples of how to reference a value from each of the tables.

SWMAN_STEP

```
key="swmanStep.comments"
```

For more information on the list of available data source values, refer to the DS_STEPS datastore documentation. Additional project-specific columns can be added to this table. The mapping for these columns is configured in the eclipselink-orm.xml file. See Configuring Project-Specific Columns on page 22-7 for more information.

Device Conditions

Web Switching has an optional Steps column that can be added to display a list of conditions that were on a device when the switching step was added. The conditions list will not dynamically update after the step is added and will only be populated when the step is recorded or manually added through the Manual Step Addition form.

To add enable the population of a conditions column in the Steps table, the following configuration changes will be required.

1. [project]\jconfig\override\fwserver.jar\META-INF\eclipselink-orm.xml:

   A new step extension field with the name 'DEVICE_CONDITIONS' will need to be added to eclipselink-orm.xml file within the SwmanStep and SwmanStepView entities.

   ```
   <basic name="DEVICE_CONDITIONS" access="VIRTUAL" attribute-
     type="String">
       <column name="DEVICE_CONDITIONS"/>
   </basic>
   ```

   The population of the device conditions field is triggered based on the field name, so the field name cannot be altered. See Configuring Project-Specific Columns on page 22-7 for more details on creating a project specific version of the eclipselink-orm.xml file.

2. [project]\sql\[project]_retain_web_swsheets.sql

   Add a new character column called 'DEVICE_CONDITIONS' to the SWMAN_STEP table.

3. Update the JBot table definition for the Steps table and add the new column. Changes will be required in files:
   - [project]\jconfig\ops\webswitching\properties\SwmanSteps_en_US.properties
   - [project]\jconfig\ops\webswitching\xml\SwmanSteps.xml

Device Attributes (Addresses)

Switching Sheet Steps

One specialized capability that the switching steps have not related to step execution is the ability to update device attribute information. When the command SaveAttributesCommand is called with a switching step extension field name, the value updated in the step for that device is propagated to the other steps in the steps list and is also passed to the device's associated attribute table. From this point on, when the device is used to record switching steps, the newly saved attribute information is displayed. In Product configuration, we utilize this feature for device address information, which is normally stored in the LOCATION field of the attribute tables. The location data is accessed through the database view ATT_SUPPLEMENTAL_ATTRIBUTES. This view is model specific and has to be defined by each project. Here is an example of the view, which should be placed into the projects sql/[project]_schema_supplemental_attributes.sql file:

```
CREATE OR REPLACE VIEW att_supplemental_attributes
(h_cls,h_idx,active,location,substation,voltage,feeder_name)
AS
  (SELECT a.h_cls,
    a.h_idx,
    a.active,
    TO_CHAR(a.location),
    TO_CHAR(a.control_zone_4),
    TO_CHAR(a.gmd_voltage),
    TO_CHAR(NVL(a.feeder_id_1, a.control_zone_5))
  FROM att_breaker a
  WHERE a.active IN ('Y', 'A')
  UNION
  SELECT a.h_cls,
    a.h_idx,
    a.active,
    TO_CHAR(a.location),
    TO_CHAR(a.control_zone_4),
    TO_CHAR(a.gmd_voltage),
    TO_CHAR(NVL(a.feeder_id_1, a.control_zone_5))
  FROM att_transformer a
  WHERE a.active IN ('Y', 'A')
  UNION
  SELECT a.h_cls,
    a.h_idx,
    a.active,
    TO_CHAR(a.location),
    TO_CHAR(a.control_zone_4),
    TO_CHAR(a.gmd_voltage),
    TO_CHAR(NVL(a.feeder_id_1, a.control_zone_5))
  FROM att_elbow a
```

```
WHERE a.active IN ('Y', 'A')
UNION
SELECT a.h_cls,
  a.h_idx,
  a.active,
  TO_CHAR(a.location),
  TO_CHAR(a.substation_name),
  TO_CHAR(a.nominal_voltage),
  TO_CHAR(a.feeder_id_1)
FROM att_generator a
WHERE a.active IN ('Y', 'A')
);
```

Each project should add any additional device types that are configured to be included in recordable device operations.

When defining a device attribute extension field for a step that should be automatically populated via the ATT_SUPPLEMENTAL_ATTRIBUTES database view, specify the extension name in lower case form. Mixed and upper case extension field names will not work.

The model attributes updated by Web Switching will be removed each time the attribute is updated from the GIS. This update can be setup to be ignored if a GIS update comes through with the old attribute value. In other words, we retain the attribute update from Web Switching as long as the GIS attribute value coming in is different. For more information, see chapter Building the System Data Model.

Switching Sheets and Safety Documents

Device attributes can also be populated into switching sheets and safety documents. The database view called ATT_SUPPLEMENTAL_ATTRIBUTES is used to populate these fields. This is the same view used to populate attribute values into switching sheet steps. In order for the JBot tool to include the attribute into a text field, the attribute has to be included as a column in the ATT_SUPPLEMENTAL_ATTRIBUTES database view.

To include attributes into a JBot text field, use the command UpdateAttributeValuesCommand. See the command's documentation for further details. With this command, a text field can be configured to be populated with a distinct comma delimited list of attribute values. For Product configuration, the values are pulled using the sheet device, step devices and for safety documents, the safety document devices.

The command can be used in any JBot tool, so the functionality is not limited to just Web Switching and Web Safety.

The ATT_SUPPLEMENTAL_ATTRIBUTES database view can also be setup as a database table and can be populated and maintained like any other attribute table. It is up to the project to determine which method is easier to maintain. The project's modeling personnel would have to be involved if the view is converted to a table. Product also includes a script to help with the initial definition of the ATT_SUPPLEMENTAL_ATTRIBUTES database view. See script `nms-generate-supplemental-attributes-view` and configure the header section of the script before running the script on your project. The script will output the statements needed to be included in the project file sql/[project]_schema_supplemental_attributes.sql. Read the comments in the header section of that script for more details. The script should be used to generate a baseline version of the view. It may not generate a perfect version of the database view and further updates should be made to the generated SQL instead of trying to update the script.

SCADA Auto-Transitioning

You can enable or disable auto-transitioning of SCADA switching steps using the `AutoTransitionSCADASteps` property, which is a property found in the `SwmanParameters.properties` file. If auto-transitioning is allowed, as it is in the product configuration, the property is set to true:

```
AutoTransitionSCADASteps = true
```

To disable auto-transitioning of SCADA switching steps, set the property to *false*:

```
AutoTransitionSCADASteps = false
```

View Areas

Each switching sheet creates a Dynamic View Area that is created the first time the sheet is saved. The default Auto-load viewer for this view area is "None." If your project would like the dynamic view area to be auto loaded into a particular viewer each time the switching sheet takes focus, then set the following parameter in each of the Swman<Sheet Type>Tool.xml configuration files. For instance, set the following property in SwmanPlannedTool.xml.

```
<StringProperty name="view_area.auto_load_unset_value"
     value="VIEW;0;2"/>
```

The view_area.auto_load_unset_value property should be assigned one of the valid key values as defined for the CMB_VIEWER_NAME combo box configured in the TBL_VIEW_AREA JBot table. For Product configuration, the possible values include VIEW;0;1, VIEW;0;2 or none. By default "none" is used if the property is not defined.

Toggling Study Mode

Each sheet type can be configured with one or more Viewers that track the mode change. This is configured with the "switching.connected_viewers" property. For product, this is configured to just track with Viewer1, but, for example, you can change your SwmanPlannedTool.xml to make both Viewer1 and Viewer2 toggle into and out of study mode along with your sheet:

```
<StringProperty name="switching.connected_viewers"
value="Viewer1,Viewer2"/>
```

Step Order Execution Rules

Each switching sheet type can be configured to force in-order step execution rules. When the rule is enforced, the Instruct, Complete, Abort and Fail options will only be enabled when all the steps prior to the selected step have been Instructed, Completed, Aborted or Failed. The same rule can also be applied for grouped steps. Normally, the sheets are configured so that grouped steps have the opposite rule applied to them so that users have the option of having some steps following the rule and others more relaxed.

The two rules are each defined in the Swman<Sheet Type>Tool.xml files. They are:

```
<BooleanProperty name="out_of_order_execution" value="false"/>
<BooleanProperty name="group_out_of_order_execution" value="true"/>
```

out_of_order_execution: When set to true, steps in a switching sheet can be Instructed, Completed, Aborted and Failed in any order. For Product configuration, this is set to True for Emergency switching sheets. Planned, Outage Correction and Template sheets have this option set to false.

group_out_of_order_execution: When set to true, steps in a step grouping can be executed out of order. If out_of_order_execution is set to true, then any steps listed prior to a

grouping have to be completed before these steps can be completed out of order. For Product configuration, this option is set to true in Planned and Template sheets. It is set to false in Emergency and Outage Correction sheets.

To alter the step execution rules for a sheet type, simply alter these values in your project version of the `Swman<Sheet Type>Tool.xml` file.

Instructed Actions

Web Switching and the Control Tool have options to treat instructed actions as if they are completed when performing a new instruct action. When configured, instructed steps in the active sheet will be considered as completed when computing the Look Ahead and associated dialogs.

For example, say you recorded an Instruct Tag step. With this option configured, you would be allowed to record an Instruct Remove Tag step with no errors. Without this configuration, you would receive an error that you cannot remove a tag that is not already applied.

This is useful for projects that instruct multiple steps to a crew at a single time. This is the default behavior for product configuration.

This configuration resides in two places, since each governs a slightly different aspect of this behavior. If you wish that instruct actions validate against the current active model, you will need to unconfigure each of these two pieces. This is not recommended unless your project instructs one step at a time.

1. In CONTROL_ACTIONS.inc, the CollectInstructedStepsCommand call in the ACT_BEGIN_ACTION Action tells the Control Tool to consider instructed steps in the current sheet. This means that an active sheet that is recording in the same mode as the Control Tool will change the Look Ahead and all associated warnings and errors to display results as if the instructed steps were already completed. This only occurs when instructing an action in the Control Tool. Remove this Command if you do not desire this behavior.
2. In SwmanStepsHeader.xml, the "apply_instructed_steps_to_lookahead" option is passed as true to the ExecuteStepsCommand so that the sheet's instructed steps affect the Look Ahead and associated dialogs, as above. This is only valid for "instruct_step" actions. Change this to false if you do not desire this behavior.

Configuring the Sensitivity of Step Execution Buttons

The DS_SWITCHING_DEFAULT.VALID_ACTIONS data element contains a list of all actions names that are available for the selected step(s) but not for any previous steps. For example, for a sheet that uses in-order execution, you would want the **Complete** button to be enabled when the selected step(s) are the next steps that can be completed. So, DS_SWITCHING_DEFAULT.VALID_ACTIONS would need to contain:

```
"SWS-complete_step":
when="{DS_SWITCHING_DEFAULT.VALID_ACTIONS == 'SWS-complete_step'}"
```

The DS_SWITCHING_DEFAULT.REVERSE_VALID_ACTIONS data element contains a list of all actions names that are available for the selected step(s) but not for any subsequent steps. For example, for a sheet that uses in-order execution, you would want the **Uninstruct** button to be enabled when the selected step(s) are the last steps that can be undone. So, DS_SWITCHING_DEFAULT.REVERSE_VALID_ACTIONS would need to contain:

```
"SWS-uninstruct_step":
when="{DS_SWITCHING_DEFAULT.REVERSE_VALID_ACTIONS == 'SWS-
uninstruct_step'}"
```

Switching Sheet Email Attachment Configuration

Web Switching Management's default behavior when emailing a switching sheet is to automatically name the switching sheet and attach the file to a new email message. The default naming convention is:

- *{sheet type}_{sheet index}.{report format}*

  Examples:

  - Planned_1003.pdf
  - Emergency_1004.rtf

An alternative configuration option allows you to provide the user with a **Save As** dialog to name the switching sheet prior to attachment to the message. To implement this option, you will need to change the value of the `$SKIP_SAVE_DIALOG$` parameter from *`true`* to *`false`* in any of the files that you wish to turn the **Save As** dialog on for:

- SwmanPlannedTool.xml
- SwmanEmergencyTool.xml
- SwmanOutageCorrectionTool.xml
- SwmanTemplateTool.xml
- SwmanTrainingTool.xml
- SwmanVoltVarTool.xml
- SwmanStandAloneSafetyTool.xml
- SwmanLoadShedTool.xml

To force the **Save As** dialog for all the sheet types and for stand alone safety documents, set the `$SKIP_SAVE_DIALOG$` parameter from true to false in the following files:

- SwmanToolBar.xml
- SwmanStandAloneSafetyMenuToolBar.xml

Automatic Crew Population Option for Steps

As part of standard Product configuration, the step's **Instructed To** field is populated by selecting a step followed by a crew from the switching sheet's Crews list. This method can also be used to pre-assign steps to crews. However, this may not line up with the business process of every project. In some cases, it may be more appropriate to automatically populate the steps based on the crews assigned to the switching sheet.

> **Note:** If this is a requirement for your project, then review the manual migration details for bugs 25469645 and 26101215.

Web Safety

State Transitions

State transitions for the safety documents are all configured in the TE State Transition database tables where the *app* value to each of the tables is set to *SF*.

> **Note:** See tables te_valid_states, te_status_groups, te_statuses, te_state_transitions, te_state_actions, te_expressions, te_init_state_rules, te_state_callbacks, and te_state_cb_args for more information.

Web Safety supports the following callbacks:

Callback Action Name	Description
check_safety_crew	Determine if a crew has been assigned to the document. This check is optional and can be configured to cause the transition to fail if a crew is not assigned. For Product configuration, we also have a validation rule setup to do the crew check on the client. This is done by calling the validation group `CHECK_FOR_DEVICES_AND_CREWS`.
check_safety_crew_position	Validate that the crews are in one of the specified `VALID_POSITION` positions. Also transition them to the `NEW_POSITION`. An unlimited number of `IGNORE_POSITION` values can also be configured. These positions will be considered valid and will not be changed to the NEW_POSITION value. None of the arguments are required; they can be used separately.
update_safety_conditions	This action is used to update the status of the conditions associated to the safety document. This action requires an argument called `STATUS`. The status argument takes a number value. A status value of zero returns the condition back to normal so that it can be manipulated by the Control Tool. The condition cannot be removed when its status is in anything other than status zero. The status value of the condition can be used to change the symbol of the condition within the viewer. An optional argument, `TAG_INFO_SCRIPT`, can be used to populate the 'safety_info' attribute in the updated conditions. This argument should be set to a JavaScript code fragment that calculates the desired value. This code is passed a copy of the SafetyDocument object as a 'doc' variable.
validate_delegated_devices	Determine if the devices listed in the Delegated Zone (DCZ) document potentially isolate a section of the network.
delegate	Delegate the DCZ document to a crew. Arguments: • CREW_NAME (required) - the field that maps to the crew data. Product uses "crew.crew_id", but other projects may map to an extn field, if this name is to be user-entered. • MUST_ISOLATE_FIRST (default: false) - whether the devices need to be open and isolated for the delegation to succeed.

undelegate	Release the delegated zone.

The following is an example for the Issue state:

```
INSERT INTO te_state_callbacks
  (app, cb_key, state_key, condition, action, abort_on_fail,
error_code)
VALUES
  ('SF', 100, 110, 'PRE_ENTER', 'check_safety_crew', 'Y', -120);
INSERT INTO te_state_callbacks
  (app, cb_key, state_key, condition, action, abort_on_fail,
error_code)
VALUES
  ('SF', 110, 110, 'PRE_ENTER', 'update_safety_conditions', 'Y', -
100);
INSERT INTO te_state_cb_args
  (app, cb_key, arg_key, arg_name, arg_value)
VALUES
  ('SF', 110, 100, 'STATUS', '1');
INSERT INTO te_state_cb_args
  (app, cb_key, arg_key, arg_name, arg_value)
VALUES
  ('SF', 110, 200, 'TAG_INFO_SCRIPT',
   'doc.getDocType() + ''-'' + doc.getDocIdx()');
```

The error_codes are used to display distinct dialog messages to the user when the action fails. The messages for these error codes are configured in the MessageCode_en_US.properties file. Here is an example for error code "-120", which was referenced in the above te_state_callbacks example.

```
OmsClientException.SF.STATE.CALLBACK.120 = Safety document has to be
assigned to a crew
OmsClientException.SF.STATE.CALLBACK.120.title = State Transition
Failed
```

The `STATUS` values define what symbols are displayed in the viewer and also define whether the condition can be removed or not. Status values from 1 to 999 will restrict the condition from being able to be removed. In order for this rule to apply, the condition must inherit from parent class `safetytag`.

The '`safety_info`' attribute in conditions associated with a safety document can also be populated by the StateTransitionCommand and SafetyConditionsCommand commands. See the documentation for these commands regarding the 'tag_info_data_source' parameter.

Safety Document Data Fields

Data fields in this case are in reference to the fields found on the safety document that are not being pulled from the associated switching sheet. Data fields can be found anywhere on the safety document.

The following is an example of how to reference these values:

```
data_source="DS_SAFETY_DOCUMENT_LOCAL.doc.extension.DESCRIPTION"
```

For more information on the list of available data source values, refer to the DS_SAFETY_DOCUMENT datastore documentation.

Search Safety Document

The Search Safety Document list is populated through the DS_SEARCH_SAFETY_DOCUMENT_LIST, which is populated from the SWMAN_SAFETY_DOCS_LIST database view. The amount of data displayed in this list will be limited to 1000 entries.

The GUI and command calls are configured in file DLG_SEARCH_SAFETY_DIALOG.xml.

The Search Safety Document list is initiated by selecting **Safety Document Search...** from the **Open** option in the Web Workspace **File** menu. The command that requests the search results is SearchSwitchOrSafetyCommand. This command has a parameter called "search_fields" that limits the number of fields to search in within the SWMAN_SAFETY_DOCS_LIST database view. It is strongly advised that your project keep this list to a minimum since the more fields to be searched, the more intensive the query will be on the system.

Configuring Stand Alone Safety Documents

For the most part, stand alone safety uses the existing configuration from the standard safety documents that are associated to switching sheets. There are a few additional configuration items that require additional attention when dealing with stand alone safety documents.

Safety Control Actions

The configuration table **SWMAN_SAFETY_TYPE_ACTIONS** plays a critical role in distinguishing key aspects that should only pertain to stand alone safety documents and not to regular safety documents associated to switching sheets. The new columns that were added to this configuration table include:

- **STAND_ALONE** - This indicates whether the action is allowed in a stand alone safety document. If a value other than 'Y' is specified for the action record, then the action will generate an error if the user attempts to paste this type of action into the stand alone safety document. During control tool recording, if the Real Time action being recorded is not found in this list, then no error is generated and the action is sent to the Miscellaneous log. Open and Close actions are prohibited from being associated to stand alone safety documents. Stand alone safety documents do not have event lists and cannot process event related processing.
- **ACTION_MAP_NUMBER** - This number is used to map key actions when safety documents are copied. For instance when copying a HOLD document as a Clearance document, the CONDADD:hold actions are all replaced with CONDADD:clear actions. This field defines that mapping in how the actions are replaced.

Flagging Safety Modifications

Stand alone safety uses a status flag called SAFETY_EDITED. This flag is defined in the SwmanParameters.properties file under the parameter **STAND_ALONE_SAFETY_EDIT_FLAG**. This flag determines when the safety documents tab label should be italicized. This flag should be set when a field on the safety document is edited by the user. This can be done like the following:

```
<TextAreaBehavior rows="4"
data_source="DS_SAFETY_DOCUMENT_LOCAL.doc.extension.NON_MODELED_DEVICE
S"  modify_flag="SAFETY_EDITED">
```

High Level Messages

SwService

The Switching Service (SwService) is used to process Web Switching Sheets that are generated or updated by the services. It accepts the following High Level Messages:

```
Action any.SwService <command> <arguments>
```

Where:

Command	Arguments	Description
debug	<N>	Sets the debug level: • 0 = Debug off • 1 = Debug on • 2 = Further details about database queries • 3 = Full debug
relock [Sheet Handle]		When no argument is given, then unlock all the switching sheets and send a request to each of the clients asking them to reestablish their single user switching sheet locks. This command can be used to clear up any orphaned locks that may still be active after an application lost network connectivity or crashed. When a switching sheet handle in the form of "<Sheet Cls>.<Sheet Idx>" is given, then only that one sheet is unlocked.

Web Switching and Web Safety

The Web Switching client application utilizes High Level Messages to receive broadcast messages. These messages are configured in the SwmanPerforms.inc file.

Web Safety messages are configured in the SafetyStartup.xml and SafetyTool.xml files. SafetyStartup.xml configuration is used by safety documents linked to switching sheets and for stand-alone safety documents; SafetyTool.xml is only used by safety documents linked to switching sheets.

The syntax for configuring HLM message receivers in the Web Switching and Web Safety tools should look like the following example.

```
<Perform name="HLM" category="onMessage"
 type="HLM_SOME_SPECIAL_MESSAGE">
    <Command value="SomeCommand"/>
</Perform>
```

From an external system or from the command line, the following action can be made to send the message out to every switching sheet or safety documented registered to listen for this message.

```
Action -java any.any HLM_SOME_SPECIAL_MESSAGE
```

To send the message to just the switching sheets, use the following command options.

```
Action -java any.swSheet HLM_SOME_SPECIAL_MESSAGE
```

Troubleshooting

Through high-level Action messages debug can be turned on or off for parts of Web Switching Management. The debug categories can be used to debug configuration issues as well as runtime issues.

Web Switching Management debug category names:

Category Name	Debug Description
DELEGATE	Delegation actions, including validating isolation devices.
HLMESSAGE	Not just for Web Switching, but this debug category displays debug about High Level message processing.
IMPACTED_CUSTOMERS	Impacted Customers.
LOCK_OBJECT	Sheet Locking and Unlocking.
SAFETY	Safety documents.
SHEET	General debug category wrapped around most actions pertaining to a switching sheet.
SHEET_EVENT_ASSOC	Event associations.
SHEET_REVISION	Switching sheet revisions.
STEP	General debug category wrapped around most actions pertaining to a single switching step.
STEP_EXECUTE	Step Executions.
STEP_REVISION	Switching sheet step revisions.
STEPS	General debug category wrapped around most actions pertaining to the switching steps.
STEPS_EDIT	Step editing.
VALIDATION	Validation rules.

Note: The server logging will show up in the WebLogic log file.

To turn on debug for a category:

```
Action any.publisher* ejb debug <Category Name>=1
```

To turn off the messages:

```
Action any.publisher* ejb debug <Category Name>=0
```

If you require only client based debug, then follow the following format:

```
Action any.publisher* ejb client <Login ID> debug <Category Name>=1
```

To turn on and off debug for all categories:

```
Action any.publisher* ejb debug 1
Action any.publisher* ejb debug 0
```

Or

```
Action any.publisher* ejb client <Login ID> debug 1
Action any.publisher* ejb client <Login ID> debug 0
```

Summary of Java Action Commands

These commands are done by using the Action command in this format:

```
Action any.publisher* ejb <command> <arguments>
```

Command	arguments	Description
debug		Enables debug for the particular class or facility. See Troubleshooting on page 22-49 for details/
dump	<type>	Dumps information about various caches to the managed server log file. type: zones = Dump zone information crews = Dump crew information events = Dump Events Information conditions = Dump condition information. cache = displays memory usage of variuos caches all = Dumps all of the above
enable_login	<true|false>	Enables or disables login. If this is set to false, it will force all users off the system
refresh		Refreshes all WebLogic cache
refresh_jms		Forces clients to refresh the JMS queue.
reload_symbology		eloads the viewer symbology configuration. Note that this only changes the server configuration. For client to pick up the new version, they need to be restarted.
stop_server		Stops the WebLogic managed server.

Chapter 23

NMS Training Simulator Configuration

This chapter describes how to configure the NMS Training Simulator. This chapter includes the following topics:

- Environment Configuration
- Power Flow Emulator Configuration
- JBot Configuration
- SRS Rules

Environment Configuration

The following must be configured for the training simulator to be used.

1. The "nms_training_simulator" product must be licensed in [project]_licensed_products.dat. See Verify Licensed Products File on page 8-9. Note that nms-setup must be run after changing [project]_licensed_products.dat.
2. The WEB_envType parameter must be set to "training" in [project]_parameters.sql. This is what differentiates a training NMS environment from other types of NMS environments (such as production and test). This parameter should NOT be set to "training" in a production environment. See "Web Application Configuration" in the *Oracle Utilities Network Management System Installation Guide*.
3. The Trainer user type must be configured. This user type must have code_script='trainer' in the ENV_CODE table.
4. Services and the Managed server must be restarted after these changes have been made.

Power Flow Emulator Configuration

The Power Flow Emulator is a separate service (EMService) that runs within NMS and submits power flow solution data back into the NMS to emulate SCADA data that would generally come from the field. The emulator is responsible for simulating SCADA telemetry, creating simulated faults events, and properly simulating user control actions that are initiated from the NMS.

The following are required to properly set up EMService:

1. PFService must run reliably on the data model.
2. The system.dat file must be setup properly to enable EMService. Ensure that the following line is setup properly.

   ```
   program EMService EMService -dbname pf -scadaId RTAdapter -inputDir
   ${NMS_SCADA_SCAN_FILE_DIR} -outputDir ${NMS_ROOT}/scada_out
   ```

   - **scadaId**: This parameter should be populated with the ID to emulate measures for.
   - **inputDir**: This parameter should be populated with the file path where RTAdapter is scanning for incoming telemetry.
   - **outputDir**: The parameter should be populated with the file path in which outbound control actions are written to.
3. The RTAdapter command line options in the system.dat file need to be changed to enabled the `-controls` parameter:

   ```
   program RTAdapter RTAdapter -scada RTAdapter -interval 3 -operate
   -dir ${NMS_SCADA_SCAN_FILE_DIR} -controls rti-emulator
   ```

4. The EMULATOR_FREQUENCY, EMULATOR_VOLTAGE_RANGE, and EMULATOR_CURRENT_RANGE SRS rules need to be set up properly to control the emulation range of values and the frequency run.

EMService High Level Messages

High level messages are available for troubleshooting and simulating user specific conditions.

> **Note:** Oracle recommends that you check with Oracle Support before using these options for the first time (especially in a production environment). They could have unintended consequences for application functionality, performance, or environment resource consumption.

- **clearoldforecastdata:** Clear any old forecast data in the DB.
- **set_emulator_options <scale factor> <day type> <hour> [<season>]:** Change the load options when running with -emulator.
- **clear_emulator_options:** Reset the load options when running with -emulator
- **set_measure_quality <handle> <attr> <quality>:** In the emulator, set the quality code device/attribute measurement.
- **clear_measure_qualities:** In the emulator, clear all measurement qualities back to zero.
- **dump_measure_qualities:** In the emulator, show all measurement quality mappings
- **simulate_op <operation> <handle> [<phases>]:** In the emulator, simulate a device operation.

- **emulate_measures:** Force the emulator to generate measurements for PFService.
- **periodic_solution:** Force the emulator to run a periodic resolve based on the state of the model.

JBot Configuration

General User Environment

When a Trainer logs in, both the USER_ADMIN and USER_TRAINER status flags get set so the Trainer will have the all of the functionality of the Admin user type plus anything configured to check the USER_TRAINER status flag. Configuration files that override JBot configuration specifically for the Trainer user are placed in the trainer directory (standard product files are in `$NMS_BASE/dist/baseconfig/product/trainer` and project-specific configuration is placed in $NMS_CONFIG/jconfig/trainer).

Switching

The Training Scenario switch sheet is configured in SwmanTrainingTool.xml. This file sets the TRAINING_SCENARIO status flag, which is used to enable functionality specific to training scenarios such as the Training menu and Offset Time column in the steps.

In various places where switch sheets are listed, filters are enabled to hide Training Scenario sheets from users that do not have the USER_TRAINER status flag set. If these configuration files are overridden by project configuration, the filters need to be included to prevent non-trainer users from seeing and possibly opening these sheets. This configuration is done in the following places: DLG_SEARCH_SWITCHING_DIALOG.xml, DLG_NEW_OR_OPEN_SHEET_DIALOG.xml, DLG_NEW_NMS_DIALOG.xml, SWLIST_TOOLBEHAVIOR.inc, and DLG_COPY_SHEET_TYPE_LIST.xml.

Control Tool

Trainer Control Tool actions are listed in the CONTROL_ACT Database Table Configuration section of the Control Tool Configuration chapter.

SRS Rules

There are several SRS rules that control the behavior of the NMS Training Simulator. These are documented in the Training Simulator related rules category in the Configuration Assistant Event Management Rules.

Chapter 24

NMS Flex Operations Configuration

Configuration in Flex Operations is driven by a combination of JavaScript configuration files, which are bundled with the product and part of the source code, and JSON data files and custom JavaScript functions that can be edited on the server in order to add or modify functionality.

Overriding Configuration and NLS

Configuration is modified in Flex Operations by customizing files on the server. The standard product configuration, Oracle's National Language Support (NLS), and some example files can be found bundled with the release under:

```
dist/baseconfig/product/flex
```

The `flex` folder contains the following files and folders:

- **`config` (folder):** All of the configuration files can be found in this folder.
- **`nls` (folder):** All of the files containing strings used in the client can be found in this folder.
- **`exampleCustomConfig.jsonc`**: An example configuration file showing how to override various elements in the configuration.
- **`exampleCustomFunctions.jsonc`**: An example JavaScript file showing how to use custom functions in the Flex Operations client. Contains some functions that are referenced in `exampleCustomConfig.jsonc`.

The folder structure beneath `config` and `nls` indicate the path required to override them, and the files themselves can be inspected to see what can be changed.

Configuration and NLS is overridden by modifying files in a specific folder on the server, which is determined by the server configuration.

```
$OPERATIONS_MODELS/config
```

This will be explained in more detail in the sections that follow.

Flex Operations and OMA Shared Configuration

OMA and Flex Operations share some configuration and strings in the map module and the product reference configuration can be found in these folders:

```
dist/baseconfig/product/flex/config/shared
dist/baseconfig/product/flex/nls/shared
```

All Flex Operations-specific configuration and NLS can be found in these folders:

```
dist/baseconfig/product/flex/config/client
dist/baseconfig/product/flex/nls/client
```

Overriding Configuration

Common configuration and NLS can be modified by adding a `config.jsonc` file in the `$OPERATIONS_MODELS/config` folder containing JSON data mirroring the folder structure in the product configuration/NLS files and the structure of the JavaScript objects defined in those files. Most Flex Operations specific configuration files can be modified by either editing the `config.jsonc` file or by overriding individual files, for example, the Work Agenda tool configuration files can be overridden by adding overrides in

`$OPERATIONS_MODELS/config/client/tools/workAgenda.jsonc`

The comment at the top of the product configuration file indicates the files where the configuration can be overridden:

```
/**
 * You can modify these configuration options by adding overrides to either the
 * main config/config.jsonc file or config/client/tools/workAgenda.jsonc
 */
```

If there is not comment like the one above, then it must be overridden in the main `config.jsonc` file.

Note: All NLS must be overridden in config.jsonc

The `exampleCustomConfig.jsonc` file gives a number of examples of how to override some specific configuration. The exact method used to override depends on the type of data being overridden in the JavaScript configuration file, which will be described in more detail in the following sections.

Functions

For functions we support a single `functions.js` file, the functions of which can be referenced from the configuration JSON files where specifying functions is supported.

When overriding product configuration, functions in the base configuration can be overridden by specifying the string name of the function in the functions.js file. When the custom function is invoked this can be used to refer to the configuration object, so it has access to the other configuration, just like the base product implementation does. It also has access to the base product implementation of the function it has replaced via this['super.'<original function name>], so the custom function can work like a class override and retain the original implementation with some tweaks, if desired.

These functions can be referenced in layout configuration where methods are specified, for example:

```
{
    "type": "button",
    "actionFunction": "common.myCustomButtonClickHandler”
```

They can also be referenced in expressions such as enabled/disabled checks or custom column definitions:

```
    "expression": "$common.myCustomFormatter(field1, field2)"
```

Observables

The system handles observables in the configuration by ensuring that if an observable or observableArray is overidden that it is updated with the new values from the overrides, not replaced.

Merging or Replacing Array/Object Content

If an array or object within a configuration class is overridden it can also be fully replaced (default behavior), or it can be merged with the array/object in the base product configuration.

Arrays

For arrays you can specify an object with a key attribute specifying the key field to match against, and up to three other arrays - add, remove and replace, containing details of array elements to add, remove or replace, for example:

```
"searchMatchTypes": {
    "key": "value",
    "add": [
        {
            "value": "Wildcard After",
            "label": "Wildcard After replaced by add"
        },
        {
            "value": "Example added by add",
            "label": "Example added by add"
        }
    ],
    "replace": [
        {
            "value": "Wildcard Before",
            "label": "Wildcard Before replaced by replace"
        },
        {
            "value": "Example added by replace",
            "label": "Example added by replace"
        }
    ],
    "remove": [
        {
            "value": "Wildcard Before and After"
        },
        {
            "value": "Remove something that doesn't exist"
        }
    ]
}
```

> **Note:** If you specify an element to add, but it is already there, it will be replaced, but a warning will be issued in the console log. Likewise, if you specify an element to replace, and it is not found, it will be added, but a warning will be issued in the console log. If you specify an element for removal, but it is not found, a warning will be issued in the console log.

Objects

If you specify the field mergeConfigValues as true in an object, the specified attributes will be added/replaced in the existing object. For example, this will replace the single "Transformer" attribute in the "mapSymbolRotation" configuration, while leaving the other attributes alone:

```
"mapSymbolRotation": {
    "mergeConfigValues": true,
    "Transformer": 180
},
```

Overriding NLS

Similar to the configuration system above, the content of the NLS files can be overridden via JSON defined under the “nls” entry in the `config.jsonc` file. The path beneath this entry can be determined by the folder structure and the JSON content of the file. The example given in `exampleCustomConfig.jsonc` is the authority tool. The authority NLS file is found at:

```
dist/baseconfig/product/flex/nls/client/tools/authority.jsonc
```

and it is overridden with this path:

```
"nls": {
    "client": {
        "tools": {
            "authority": {
                "columns": {
                    "zoneName": "Name"
                }
```

Refer to the product configuration files to see what strings there are and the path to override them. In the example above, the columns are defined like this:

```
{
    "columns": {
        "zoneName": "Zone Name",
        "user": "Users",
        "ruleset": "Rule Set",
        "omaSelfAssign": "OMA Self Assign",
        "suppress_callbacks": "Callbacks",
    },
```

Common Terms/Concepts

In order to account for different regions having different terms for the same thing, certain terms/concepts are referenced via some common NLS so they can be easily replaced everywhere in the system and do not have to be laboriously replaced everywhere they appear.

Common terms are defined in an NLS file just like other strings, and can be overridden as shown above, under the 'common' path. These common strings are referenced in the NLS as embedded variables using the same format as other NLS substitutions, for example:

```
{
    ...
    menus.editEvent = 'Edit {common.event}',
    ...
}
```

Unlike other NLS strings, the key of the common term can be capitalized to capitalize the end result, for example, 'Edit {common.Event}' will become 'Edit Event'. This avoids having duplicates of all the capitalization options in the common NLS.

An example of overriding common NLS terms can be found in `exampleCustomConfig.jsonc`:

```
    "nls": {
        "common": {
            "an_event": "a job",
            "event": "job"
        },
```

Note on Grammar Issues

There are some issues around grammar when doing this, so in addition to event, in the common terms above we also have a_event, defined as "a**n** event". This is because if we wanted to call events jobs, we couldn't just replace event with job, as "a**n** job" is not grammatically correct. Having an event defined as "an event" allows it to be changed to "a job" and so the substitution works grammatically.

Similarly we have openSwitch and openedSwitch, to account for the instruction versus the state.

Testing Changes

To test changes made to configuration files simply reload the Flex client. You may need to force the browser not to cache the files as sometimes browsers are reluctant to check for new versions of files they have recently downloaded. The most reliable way to do this is to use the browser development tools – under the network tab there will be an option to disable the cache – enable this option and refresh the page and you should immediately see any changes reflected in the UI.

Layout Configuration

In general, Flex Operations tools are broken into separate ElementDefinition JSON blocks, like a toolbar, body, separate tab definitions, and the containing UI definition. These are not all required; elements and sub-elements can be added as desired.

The ElementDefinition contains type, id, layout, and a list of children elements. These children are the keywords that define the fields and actions you can add to your tool, from the FieldMapping and ToolActionMapping defined elsewhere in the [tool]Config.ts file.

For example in `eventDetailsConfig.ts`, we define some fields as the header:

```
header: ElementDefinition = {
    id: 'HEADER',
    type: ElementType.Panel,
    layout: {
    type: LayoutType.Responsive
},
childNlsPath: 'fields',
```

This first part indicates that we are starting a panel with standard Responsive layout (default: `small:12`, `medium:12`, `large:12`). The strings for the children elements will be found under the “fields” element in the eventDetailsNls.ts file.

We then continue to add children:

```
children: [
{
    type: ElementType.Panel,
    layout: {
        type: LayoutType.Responsive,
        small: 12,
        medium: 6,
        large: 3
},
```

This section defines a Responsive sub-panel, but with a change to the default sizes. Responsive elements stack horizontally until the combined sizes are greater than 12, at which point the next element starts a new row. Small, medium, and large refer to the width of the browser itself. So when the browser I narrow (small), this panel will take all 12 of the horizontal slots, and the next panel will need to stack below it. When the browser is medium width, it can share a row with the next panel, and at a large width, 4 of these panels will share a row. This allows the tools to adapter dynamically to the browser width.

Now define our true child elements:

```
        children: [
            {
                id: 'EVENT_IDX',
            },
            {
                id: 'PHASES',
            },
            {
                id: 'NUM_CALLS'
            },
            {
                id: 'NUM_CUST_OUT'
            }
        ]
    },
…
```

The EVENT_IDX, PHASES, NUM_CALLS, and NUM_CUST_OUT are already defined in the FieldMapping section:

```
EVENT_IDX: {
    field: 'data.event.id',
    type: ElementType.Text,
    readonly: true
},
NUM_CALLS: {
    field: 'data.event.numCalls',
    type: ElementType.Text,
    readonly: true
},
NUM_CUST_OUT: {
    field: 'data.event.numCustOut',
    type: ElementType.Text,
    readonly: true
},
```

Similarly, you add ActionElementDefinitions for actions defined in the ToolActionMapping section:

```
   ADD_EVENT_NOTE_PANEL: <FlexBarDefinition>{
      type: ElementType.FlexBar,
      middleElement: {
         id: 'OPERATOR_COMMENT',
         layout: {
              type: LayoutType.FlexBarMiddle
         },
      },
      endElement: <ActionElementDefinition>{
         action: 'ADD_COMMENT',
         layout: {
              type: LayoutType.FlexBarEnd,
              padding: true
         },
         type: ElementType.Button
      },
   },
…
this.actions = new ToolActionMapping(this.nls.actions, {
   ADD_COMMENT: {
      iconClass: 'oj-ux-ico-plus',
      actionFunction: 'onAddComment'
   },
…
```

Custom Fields

You can define custom fields using the expression attribute in a GridDefinition table column or in a FieldMapping element.

Use the `$subst()` function to substitute using fields in the corresponding f`ormatNls.ts` file. This by default uses the base column (without the “#”), but can accept a different column name as an optional second parameter.

Specify other columns names to concatenate or add as needed.

For example:

```
    TABLE: <GridDefinition>{
…
        columns: [
            { field: '#METER_PHASES', expression: '$subst("phases")' },
            { field: '#VOLTAGE_STR', expression: 'VOLTAGE_A + "/" +
            VOLTAGE_B + "/" + VOLTAGE_C’ },
…
    <FieldMapping>: {
        CRIT_DHN_GROUP: {
            expression: 'data.event.critDCustOut +
            data.event.critHCustOut + data.event.critNCustOut',
            type: ElementType.Text
    },
```

And in the formatNls.ts file:

```
…
 phases: {
   1: 'A',
   2: 'B',
   3: 'AB',
   4: 'C',
   5: 'AC',
   6: 'BC',
   7: 'ABC'
 },
```

Enabled/Visible Configuration

Both the `ToolActionMapping` and the `FieldMapping` support the `enabled` and the `visible` attributes. In these attributes, specify JavaScript clauses that evaluate to true or false. You can use the `$ui prefix` to access UI components, then perform your checks against them.

For example, this enables the `DEVICE_PHASES` CheckBox when the `EVENT_TYPE` is set to `RESTORED`:

```
DEVICE_PHASES: <CheckBoxGroupDefinition>{
   enabled: '$ui.EVENT_TYPE.currentValue == "RESTORED"',
   field: 'data.event.phases',
   type: ElementType.CheckBoxGroup,
   options: 'data.event.phaseOptions'
}
```

This enables the `POSITION` action when the EVENTS_TABLE has a single selection and when the selected row has a device alias:

```
POSITION: {
   enabled: '$ui.EVENTS_TABLE.hasSingleSelection &&
   $ui.EVENTS_TABLE.row.device_alias',
   actionFunction: 'onPositionSelectedEvent'
},
```

This enables the `MAKE_RADIAL` action when the table has any number of selections, and they all have the radial flag set to false:

```
MAKE_RADIAL: {
   enabled: '$ui.EVENTS_TABLE.hasSelection &&
   $ui.EVENTS_TABLE.allMatch("!$row.radial")',
   actionFunction: 'onMakeRadialSelectedEvent'
},
```

Tables support the following:

- **hasSelection:** true/false
- **hasSingleSelection:** true/false
- **row:** row data for the single selection
- **allMatch()**: function that allows you to access each row using the "$row"
- **nonMatch():** function that allows you to access each row using the "$row"
- **anyMatch():** function that allows you to access each row using the "$row"

Other UI components support:

hasValue: true/false

currentValue: the object in that component selected (for Tabs): the key of the selected tab

You can also access data members, as also referenced in the FieldMapping section.

Chapter 25

Oracle BI Publisher Reports Configuration

This chapter describes how to install and configure the Oracle BI Publisher Report packages for use with the Oracle Utilities Network Management System. It includes the following topics:

- CES_PARAMETERS Configuration for BI Publisher Reports
- Installing the Oracle BI Publisher Report Packages
- Altering and/or Translating the Reports
- Contents of the WebSwitching Folder
- Contents of the WebSwitchingImpactedCustomers Folder
- Contents of the WorkAgenda Folder

CES_PARAMETERS Configuration for BI Publisher Reports

Configure the following parameters in the CES_PARAMETERS configuration table. Once the changes have been made, WebLogic will require a restart for the changes to take effect.

CES_PARAMETERS Attribute	Description
WEB_bipub.JDBCDriverClass	The JDBC driver class for the data source, for example, `oracle.jdbc.OracleDriver` (the default value).
WEB_bipub.reportPath	The URL to BI Publisher, for example, `http://bip_server:9704/xmlpserver/`. This is a site-specific parameter.
WEB_bipub.reportFolder	The chosen environment name; such as *Test*, *Training*, or *Production*. If you only plan to manage reports for one NMS environment within BI Publisher, then leave this parameter value empty.
WEB_bipub.userName	The user name for logging into BI Publisher. This is a site-specific parameter.
WEB_bipub.password	The password for logging into BI Publisher. This is a site-specific parameter.

CES_PARAMETERS Attribute	Description
WEB_bipub.locale	For translated versions of the reports, update the locale to the appropriate value for your region. For English, this value should be set to en-US.

Note: All the other WEB_bipub.* parameter values should be left empty. This includes WEB_bipub.JDBCURL, WEB_bipub.JDBCUserName, WEB_bipub.JDBCPassword and WEB_bipub.dataSourceName.

Installing the Oracle BI Publisher Report Packages

The Oracle Utilities Network Management System product configuration has a packaged version of Web Switching and Work Agenda reports that may be used to email, preview, and print switching sheets, safety documents, Miscellaneous Log, and Work Agenda call reports. The package is delivered in the `nms_configuration.zip` file, which extracts to the following folders:

- $NMS_CONFIG/jconfig/ops/bi_publisher/WebSwitching
- $NMS_CONFIG/jconfig/ops/bi_publisher/WebSwitchingImpactedCustomers
- $NMS_CONFIG/jconfig/ops/bi_publisher/WorkAgenda

Each folder contains files that must be uploaded to the BI Publisher Catalog. The next set of steps will guide you through this process.

Installation

When configuring BI Publisher for multiple environments, the CES_PARAMETER WEB_bipub.reportFolder will need to have a unique setting on each of the NMS environments.

Installation Steps

1. Log into BI Publisher (`http://<BIP server name>:9704/xmlpserver/`) as the Administrator from a browser that has access to the WebSwitching and/or WorkAgenda folder that was extracted from the nms_configuration.zip file. The WebSwitching or WorkAgenda folder can be copied from their default installation directory to a PC of your choice.
2. Set up a database connection by going to the Oracle BI Publisher Administration page, navigating to the **JDBC Connection page** under the Data Sources section, and then click **Add Data Source**.
3. In the **Name** field, enter `<reportFolder>` (*i.e.*, the value of the `WEB_bipub.reportFolder` parameter).
4. Set the **Driver Type** to Oracle 9i/10g/11g.
5. Set the **Database Driver Class** to `oracle.jdbc.OracleDriver`.
6. Set the **Connection** string to:
   `jdbc:oracle:thin:@<your machine>:1521:<the ORACLE_SID>`
7. Set the **username** and **password** to match your Oracle Utilities Network Management System database login values.
8. Click **Test Connection** and verify that it is properly configured.
9. Click **Apply.**

10. The following steps are needed when using BI Publisher 12c:

    If the Administration Superuser has not been specified in BI Publisher, then the following steps will need to be completed before reports can be uploaded into the BI Publisher environment.

    - Select the **Administration** button at the top of the page.
    - Select the **Security Configuration** link.
    - Click **Enable Local Superuser**.
    - Enter the user and password you are logging in as to do the report uploads and alterations.
    - Restart BI Publisher to apply the changes.

    Complete the following steps for uploading the report files.

11. From the **BI Publisher Catalog** page, select **Shared Folders** from the folders tree.
12. On top of the folders section, click the **New** drop down and select **Folder** from the list.
13. In the Folder Name field, enter `<reportFolder>` (*i.e.*, the value of the WEB_bipub.reportFolder parameter).
14. Click **Create**. The new `<reportFolder>` folder will be added to the Folders list. You may have to click the toolbar **Refresh** button to update the Folders list.
15. Select the `<reportFolder>` folder in the Folders list.
16. In the Tasks section on the bottom left of the page, click **Upload**.
17. Browse to the `WebSwitching` report package folder that was extracted from the nms_configuration.zip file and upload the WebSwitching.xdrz archive.

    Note: When replacing existing reports, select the **Overwrite existing report** option followed by the **Upload** button.

18. Select the Shared Folders/*<reportFolder>*/WebSwitching folder in the Folders list.
19. To the right of the Folders list on the Catalog page, select the **Edit** link for the NMS Data Model item.
20. Select **Properties** at the top of the Data Model tree.
21. Change the 'Default Data Source' from WebSwitching to the data source you created in step 3. The data source should have the same name as your `<reportFolder>` parameter.
22. Click the save button at the top right to save your changes.
23. Select the **Catalog** link at the top of the Catalog page to navigate back to the package view.
24. To the right of the Folders list on the Catalog page, select the **Edit** link for the Templates item. From the toolbar, change the data model to the newly modified NMS Data Model. You do this by selecting the **Select Data Model** button ( ) found on the toolbar. From the Select Data Model dialog, select the **NMS Data Model** entry from the Shared Folders/<reportFolder>/WebSwitching folder and click **Open**.
25. Repeat steps 15-24, replacing WebSwitching with WorkAgenda.
26. Repeat steps 15-24, replacing WebSwitching with WebSwitchingImpactedCustomers.

You should now be able to preview and print reports from the Web Switching and Work Agenda applications. If you have made any changes to the WEB_bipub CES_PARAMETERS parameters, then a restart of WebLogic will be required before the reports can be used by NMS.

Altering and/or Translating the Reports

Adding XLIFF translation file

The reports used for printing can be easily translated to alternate languages or the labels updated to something more appropriate to the project. Simply edit the report layout, open the 'Layout Properties' page and click **Extract Translation**. Within this XML file you will find a number of `<trans-unit>` elements with `<source>` and `<target>` sub-elements. Update the `<target>` entry with your translated or altered label.

For all languages except Chinese and Portuguese (Brazil), if you wish to create a language specific version of the XLIFF file, name the translated report file according to the following standard:

`WebSwitching_<language_code>.xlf or WorkAgenda_<language_code>.xlf`

where `<language_code>` is the two-letter ISO language code (in lower case). For example, `WebSwitching_en.xlf for English.`

For Chinese (China), Chinese (Taiwan), and Portuguese (Brazil) you must use the language code and territory code in the translated file name as follows:

```
WebSwitching_zh_CN.xlf
WebSwitching_zh_TW.xlf
WebSwitching_pt_BR.xlf
```

For more information on translating reports, see the section "Translating Reports" in the *Oracle Business Intelligence Publisher User's Guide*.

In order to utilize a language specific XLIFF file, the WEB_bipub.locale parameter has to be set correctly in the CES_PARAMETERS table. For example, if the language is English, the XLIFF file name would be **WebSwitching_en.xlf** and **WEB_bipub.locale** would be set to **en-US** in the CES_PARAMETERS table.

Updating the Sub-Template and Template Files

The sub-template and template files can be altered to accommodate project requirements.

From the **BI Publisher Home** page, select the **Download BI Publisher Tools** option and select the version of BI Publisher Desktop that matches the version of Microsoft Office on your PC. Once installed, use Microsoft Word to edit the RTF sub-template and template files. Labels and the layout of data entries can be easily manipulated from this editor.

A new menu named **Oracle BI Publisher** will be added to Microsoft Word. Select the **Help** option from menu for more in-depth information on editing templates.

The Work Agenda report templates were not created as RTF files and instead were developed from within BI Publisher. To edit the Work Agenda reports, simply select the Edit link from the WorkAgenda Templates item found in the BI Publisher WorkAgenda folder. This will initiate the editor built into BI Publisher. For more information on the BI Publisher Editor, see the *Oracle Business Intelligence Publisher User's Guide*.

The Web Switching Impacted Customers report template is not used by the Product export options CSV (Comma Separated Values) and XML. These output types are driven by the data model queries and do not require a report template, but an empty report template is required for the request to work properly from NMS.

Updating the Report Template File

The RTF templates use data extracted from queries defined in the BI Publisher data model. For Web Switching, the data model is defined in the NMS Data Model. The reports are configured in two template files. The Sub Template defines each section of the switching sheet to be included in the report. These sections can then be included in each individual template that defines the layout for each switching sheet type. With the sub template, you only need to make a change once and have it included by multiple templates in the final reports. If each of your sheet type templates will have completely different layouts, then the sub template sections may not be of any use. It is up to each project to determine how much configuration they will reuse in their reports.

To modify the data model and alter the data sets, do the following:

1. Log into BI Publisher 11g.
2. Go to the Catalog Folders and select the **WebSwitching** folder where your report templates and data model are stored.
3. Select the **Edit** link for the NMS Data Model entry in the WebSwitching folder.
4. On the left you will find all the data sets used by the Web Switching reports. Select one of the data sets. For example, select the Q_STEPS data set.
5. The Data Model viewer will be displayed. Click the **Edit Selected Data Set** button on the Diagram tab's toolbar.
6. The Edit Data Set dialog is displayed. From this editor, you can alter the query to include additional data elements that may be required to be referenced in the RTF templates. Click **OK** when you are finished.
7. In the upper right-hand corner of the NMS Data Model tab, click the **Save** button to save your changes.

To modify RTF template files, do the following:

Sub-templates:

1. Log into BI Publisher 11g.
2. Go to Catalog Folders, and select the WebSwitching folder where your report templates and data model are stored.
3. Select the **Edit** link for the SubTemplate entry in the WebSwitching folder.
4. From the Sub Template tab, select the English (United States) entry from the Templates list. You will be asked to open or save the file. Save the file to your local PC.
5. From your local PC, edit the file you just saved using Microsoft Word and close the document when you are done.
6. Go back to the Sub Template tab within BI Publisher and click the **Upload** button to upload the newly edited sub template file.
7. Save the Sub Template changes in BI Publisher. Initiating a print or email request from NMS will utilize your new template changes.

Report Template:

1. Log into BI Publisher 11g.
2. Go to Catalog Folders and select the WebSwitching folder where your report templates and data model are stored.
3. Select the **Edit** link for the Templates entry in the WebSwitching folder.
4. Select the **Edit** link for the report template you wish to edit. You will be asked to open or save the file. Save the file to your local PC.

5. From your local PC, edit the file you just saved using Microsoft Word and close the document when you are done.

6. Go back to the Templates tab within BI Publisher, delete the original template and click the **Add New Layout** button found on the upper right corner of the Templates folder.

7. From the Upload or Generate Layout section, select the **Upload** option. Fill out the Layout Name, File, Type (RTF), and Locale (English (United States)). The Layout Name should be one of the following: VVOSheet, EmergencySheet, PlannedSheet, TemplateSheet, OutageCorrectionSheet, TrainingScenarioSheet, MiscLog, or StandaloneSafety.

8. Save the Template changes in BI Publisher. Initiating a print or email request from NMS will utilize your new template changes.

Changing Date Formats

BI Publisher Report Templates

BI Publisher report template date fields are formatted using the `NMS_DATE_FORMAT`, which is based on the `Centricity.DBDateTimeFormat` parameter in Global_en_US.properties (product/global/properties/Global_en_US.properties).

Miscellaneous Log Report

Dates in the Miscellaneous Log report use the CentricityTool.properties `DBDateFormat` parameter, which does not include time. To alter the date format for the Miscellaneous Log, update the `DBDateFormat` parameter in Global_en_US.properties (product/global/properties/Global_en_US.properties).

Contents of the WebSwitching Folder

- oracle_sig_logo.gif: the Oracle logo used in the header of the generated report.
- WebSwitching.xdrz: an archive file that includes the entire contents of the WebSwitching folder within BI Publisher. This archive can be opened up with any archive viewer. The archive contains the following files:
  - NMS Data Model.xdmz: BI Publisher data model. This file defines the data that is used by all the switching sheet and safety document reports. It contains all the queries that are used to pull the data from the database. This includes Web Switching, Event, Crew, Customer and Web Safety information.
  - SubTamplate.xsbz: BI Publisher sub-template. This file consists of template definitions for all the common section of a Web Switching report that are called from all the Web Switching report templates.
  - Templates.xdoz: BI Publisher templates. This file consists of the templates for all the reports related to Web Switching and Web Safety. The templates included are VVOSheet, EmergencySheet, PlannedSheet, TemplateSheet, OutageCorrectionSheet, TrainingScenarioSheet, MiscLog, and StandaloneSafety.

Contents of the WebSwitchingImpactedCustomers Folder

- WebSwitchingImpactedCustomers.xdrz: an archive file that includes the entire contents of the WebSwitchingImpactedCustomers folder within BI Publisher. This archive can be opened up with any archive viewer. The archive contains the following files:
  - NMS Data Model.xdmz: BI Publisher data model. This file defines the data that is used by the Web Switching Impacted Customer export reports. It contains all the queries that are used to pull the data from the database.
  - Templates.xdoz: BI Publisher templates. This file consists of a generic report. Product is currently only configured to export CSV (comma separated value) and XML outputs. These output types do not require a report template as the data is organized based on the database queries defined in the data model. Project templates can certainly be added and additional output types be added through project configuration.

Contents of the WorkAgenda Folder

- WorkAgenda.xdrz: an archive file that includes the entire contents of the WorkAgenda folder within BI Publisher. This archive can be opened up with any archive viewer. The archive contains the following files:
- NMS Data Model.xdmz: BI Publisher data model. This file defines the data that is used by all the work agenda call reports. It contains all the queries that are used to pull the data from the database.
- Templates.xdoz: BI Publisher templates. This file consists of the templates used by the Work Agenda reports.

Chapter 26

Building Custom Applications

The intended audience for this chapter are software programmers responsible for building interfaces and applications that interact with the Oracle Utilities Network Management System. This chapter includes the following topics:

- Overview
- Prerequisites
- Compiling C++ Code Using the Software Development Kit

Overview

This chapter describes how to build C++ and Java applications that interact with the Oracle Utilities Network Management System using the Oracle Utilities Network Management System Software Development Kit (SDK).

Most Oracle Utilities Network Management System implementations will require at least one custom built application, a model interface, while other implementations may have addition interfaces and other programs that interact with the Oracle Utilities Network Management System. To support the implementation of these interfaces and programs, the Oracle Utilities Network Management System has provided a Software Development Kit. The Software Development Kit is installed into the `$NMS_BASE/build` directory and is pointed to using the .nmsrc environment variable $NMS_BUILD.

There are two subcomponents to the Software Development Kit:

$NMS_BUILD/make	The make rules to support the architecture and platform configuration.
$NMS_BUILD/include	The C++ header files required to interact with the Oracle Utilities Network Management System.
$NMS_BASE/sdk/java/lib	The jar files containing compiled Java classes required to interact with the Oracle Utilities Network Management System.
$NMS_BASE/sdk/java/docs	Documentation for the Oracle Utilities Network Management System Java API.
$NMS_BASE/sdk/java/samples	Sample Java applications. In this release, a sample MultiSpeak-based AMR or AVL adapter is included.

Note the following regarding usage of the Oracle Utilities Network Management System Software Development Kit:

- The SDK interfaces are not documented and are for use as-is.
- The SDK interfaces may change from release to release with no guarantees of forward or backward compatibility.
- The use of the SDK can impact the running Oracle Utilities Network Management System based on what is programmed with the SDK. Impacts may include performance issues, system lock ups, system instability, data loss, and changes to system functionality. It is recommend that you heavily test any interfaces or programs you create and judge the impact on the Oracle Utilities Network Management System and understand these interfaces and programs should be considered "use at your own risk".
- The SDK may not be used to reverse engineer the features and functionality of the Oracle Utilities Network Management System.

Prerequisites

The Oracle Utilities Network Management System Software Development Kit requires the following:

- GNU Make
- Apache Ant
- Java SE Development Kit
- Java EE Software Development Kit

Verify that your .nmsrc was generated using the `$NMS_BASE/bin/nms-env-config` script or based on the `.nmsrc.template` file in the `$NMS_BASE/templates` directory. The `nms-env-config` script can be used over and over again to review and/or update NMS specific environment variable configuration. Make sure the `$NMS_BUILD env variable is set to $NMS_BASE/build` template from `$NMS_BASE/templates/nmsrc.template` and that the environment variable $NMS_BUILD is set to `$NMS_BASE/build`.

Compiling C++ Code Using the Software Development Kit

Place the C++ source code to build the custom interface or program in a subdirectory of the $NMS_CONFIG directory, typically $NMS_CONFIG/apps. The executables resulting from the compile will be generated into the $NMS_CONFIG/bin directory via the Makefile so the nms-install-config process can copy them to the runtime directory, $NMS_HOME/bin. If you create custom shared libraries, these need to be copied into $NMS_CONFIG/lib so they also are available for nms-install-config to copy them to the runtime directory, $NMS_HOME/lib.

The following is an example Makefile for the $NMS_CONFIG/apps directory:

```
#########################################################################
#
# Example $NMS_CONFIG/apps directory Makefile
#
#########################################################################
# Include compiler and architecture dependent Makefile parameters.
HAS_GUI = YES
include $(NMS_BUILD)/make/make.rules
LOCALLIBS = $(PP_LIB) $(MV_LIB) $(SUPPORT_LIBS) $(MB_LIB) $(GRWINDOW_LIB)

# Source for all run-time applications
SOURCES = \
        CustomInterface.C
OBJECTS = $(SOURCES:.C=.$(OBJ_EXT))
PROGRAM = CustomInterface$(EXE_EXT)

#########################################################################
# Targets

include $(SIMPLE_PROGRAM_MAKE)

all:: $(PROGRAM)
        @ if [ ! -d "$(NMS_CONFIG)/bin" ]; then \
            mkdir $(NMS_CONFIG)/bin; \
          fi
        cp $(PROGRAM) $(NMS_CONFIG)/bin;
```

The target executable file in this example is `CustomInterface` and the C++ source code to compile is `CustomInterface.C`.

From the command prompt within the $NMS_CONFIG/apps directory, build the custom program with "make clean" to remove old compiled binaries and "make" to compile and install the binaries into the $NMS_CONFIG/bin directory.

Below is an example of what the output from the make system will look like as a result or running these two commands.

```
nms-vm:nms1> cd ~/OPAL/apps
nms-vm:nms1> make clean
rm -f *.o *~ core .pure* gmon.out  so_locations *.sl *.so *.a
rm -f \#* 3log *.third *.third.*
rm -rf ptrepository cxx_repository Templates.DB SunWS_cache tempinc
rm -f OPAL_preprocessor
nms-vm:nms1> ls
Makefile  OPAL_imp_exp.C  OPAL_preprocessor.C  OPAL_preprocessor.h
nms-vm:nms1> make onsite
OPAL_preprocessor.o
g++ -pedantic -W -Wall -Wno-format-y2k -Woverloaded-virtual -Wpointer-arith -Wca
st-align -Wwrite-strings -Wno-long-long -Wsign-promo  -g   -DDIFFUSION_NOTIFIES
   -DLINUX -D_REENTRANT -DP_THREADS   -DHAS_XT -DFAULT_RESTORATION -DGSOAP_VERSI
ON= -I/users/nms1/nms/product/1.10.0.0/build/include   -I/users/nms1/nms/product
/1.10.0.0/isis/include  -I/opt/oms-10.1/include     -c  OPAL_preprocessor.C
motif Building OPAL_preprocessor:
g++ -pedantic -W -Wall -Wno-format-y2k -Woverloaded-virtual -Wpointer-arith -Wca
st-align -Wwrite-strings -Wno-long-long -Wsign-promo  -g   -DDIFFUSION_NOTIFIES
   -DLINUX -D_REENTRANT -DP_THREADS   -DHAS_XT -DFAULT_RESTORATION -DGSOAP_VERSI
ON= -I/users/nms1/nms/product/1.10.0.0/build/include   -I/users/nms1/nms/product
/1.10.0.0/isis/include  -I/opt/oms-10.1/include     -L/users/nms1/nms/product/1.
10.0.0/lib  -o OPAL_preprocessor OPAL_preprocessor.o  -lPp  -lMv  -lMv  -lApp -L
/opt/oms-10.1/lib -lxrttable -lpdsutil -lxrttablestub -L/opt/oms-10.1/lib -lXpm
 -lCrew  -lPp  -lService  -lMB  -lGrWindow   -lintersys_xt  -lWrapper  -lBase  -
lfoss -L/users/nms1/nms/product/1.10.0.0/lib -L/users/nms1/nms/product/1.10.0.0/
isis/lib -lisisX -lisis -lisis_task_native  -lCmdLine  -L/opt/oms-10.1/lib -lMrm
 -lXm -lXp -lXext  -L/opt/oms-10.1/lib -lXt -lX11  -lpthread -ldl  -L/opt/oms-10
.1/lib  -lgsoap++  -lgsoap
Building and Linking C++ OPAL_imp_exp:
g++ -pedantic -W -Wall -Wno-format-y2k -Woverloaded-virtual -Wpointer-arith -Wca
st-align -Wwrite-strings -Wno-long-long -Wsign-promo  -g   -DDIFFUSION_NOTIFIES
   -DLINUX -D_REENTRANT -DP_THREADS   -DHAS_XT -DFAULT_RESTORATION -DGSOAP_VERSI
ON= -I/users/nms1/nms/product/1.10.0.0/build/include   -I/users/nms1/nms/product
/1.10.0.0/isis/include  -I/opt/oms-10.1/include     -L/users/nms1/nms/product/1.
10.0.0/lib   -o OPAL_imp_exp OPAL_imp_exp.C  -lPp  -lMv  -lMv  -lApp -L/opt/oms-
10.1/lib -lxrttable -lpdsutil -lxrttablestub -L/opt/oms-10.1/lib -lXpm  -lCrew
-lPp  -lService  -lMB  -lGrWindow   -lintersys_xt  -lWrapper  -lBase  -lfoss -L/
users/nms1/nms/product/1.10.0.0/lib -L/users/nms1/nms/product/1.10.0.0/isis/lib
-lisisX -lisis -lisis_task_native  -lCmdLine  -L/opt/oms-10.1/lib -lMrm -lXm -lX
p -lXext  -L/opt/oms-10.1/lib -lXt -lX11  -lpthread -ldl  -L/opt/oms-10.1/lib  -
lgsoap++  -lgsoap
cp OPAL_preprocessor OPAL_imp_exp ../bin
nms-vm:nms1>
```

Note: By default, project compiles produce debug builds. To improve performance, you can change to optimized mode by adding the following to the.profile configuration file in your compilation environment:

```
export NMS_COMPILE_OPTIMIZED=1
```

After you have successfully compiled the custom application, run `nms-install-config` to pick up the executables from the $NMS_CONFIG/bin and install them into $NMS_HOME/bin.

Building Sample AMR and AVL Adapter

The sample AMR/AVL adapter is a J2EE application, which is intended to be deployed on a J2EE application server, such as Oracle WebLogic Server.

Build Instructions

Source code for the sample adapter is located in the `$NMS_BASE/sdk/java/samples/amr` directory.

1. Edit the build.properties file:

   a. Specify the location of the NMS SDK.

   The NMS SDK is expected to be in the `$NMS_BASE/sdk` directory. The nms.sdk.dir property can be used to specify different location.

   Uncommment the nms.sdk.dir property line and add the appropriate path.

   ```
   # Location of NMS SDK.
   # If not set then NMS_BASE environment variable is used
   # to locate it.
   #
   nms.sdk.dir = path_to_the_dir
   ```

   b. Specify location of the Java EE jar files.

   - If using WebLogic application server and the MW_HOME environment variable is set, then the jar files from $MW_HOME/modules directory will be used.
   - If Java EE 6 SDK is used, then you will have to set the *javaee6.sdk.dir* parameter to point to the Glassfish installation.

   ```
   #
   # If using Java EE 6 SDK to build the adapter this property
   # should be set to the Glassfish install directory.
   #
   javaee6.sdk.dir = path_to_the_dir
   ```

2. From a command prompt, execute the following command:

   ```
   ant clean all
   ```

If the build is successful, the build file, demo.ear, will be created in the build directory.

Deployment

In order to run the sample application, the Oracle Thin JDBC driver has to be available on the application server where the adapter will be deployed.

The sample adapter uses a subset of the configuration options of the Oracle Utilities Network Management MultiSpeak Adapter. See the MultiSpeak Adapter chapter of the *Oracle Utilities Network Management Adapters Guide* for configuration and deployment instructions.

Chapter 27

OMS for Water

To convert the NMS to a Water OMS, "water" needs to be licensed in your `[project]_licensed_products.dat`. This automatically applies overrides from electrical to water terminology. Create a `[project]_overrides.sql` if any further changes are needed. Note that there are specific keywords in this and other configuration files such as **device_open**, **device_opened**, **device_close**, **device_closed**. These keywords exist to reduce any ambiguity between the word **open** when it refers to opening a device, opening a tool, or a device that is already open.

Appendix A

Command Line Options and High Level Messages

This appendix provides a reference for command line options and high level messages.

- Command Line Options
- High Level Messages

Command Line Options

This section contains the following Command Line Option categories:

- Command-Line Options Valid For All Services
- DBService
- DDService
- JMService
- MBService
- MTService
- ODService

Command-Line Options Valid For All Services

`-debug <integer>`

`-debug <facility> <integer>`

Turn on debug, either for all facilities or for the specified facility.

`-pgtiming on`

Turn on process group timing debug.

DBService

-database

Specify name of logical RDBMS.

-debugSqlTiming

Add SQL timing information to the log file. At level 1, all SQL commands (queries/updates) are printed. At higher levels, all SQL commands that take more than the specified number of milliseconds are printed. For example, -debugSqlTiming 500 prints all SQL commands that take more than 500 ms.

-forceCursorSharing

Sets the cursor_sharing option to FORCE for all RDBMS connection sessions managed by this instance of DBService.

-maxconnections [N]

Specifies the maximum number of concurrent RDBMS sessions (dbrdwr processes) to support.

-process_name [name]

Use process name other than DBService.

-queryOut

Writes queries to standard output.

-service [service_name]

Specify name of database service. Default is ops.

DDService

-alarmUncontrolledZones [number]

Activate the generation of alarms when DDService detects control zones without a controller or subscriber. This option takes an optional argument which is the rate (in seconds) at which to re-generate an alarm if the zone is still unsubscribed.

-alarms [EVENTS EXP UNEXP MEASURES MANUAL ABNORMAL SMS or ALL]

Create a new alarm for any of the following actions. Note that these options can be used in any combination and do not affect one another, although there is some occasional overlap. Also note that the **alarmDigital** SRS_RULES entry controls whether DDService creates alarms for changed non-status digitals. This is separate configuration and not included with the **ALL** option below.

- CONDITIONS all tag, note, and other condition actions
- EVENTS controls sent to the SCADA system
- EXP Centricity device operations
- UNEXP SCADA device operations
- MEASURES measurements and overrides

- MANUAL Centricity device operations (same as EXP)
- ABNORMAL Abnormal device statuses
- SMS SMService-initiated alarms (like service failures)
- ALL All of the above
- ALL is the default value, when -alarms is used with no arguments
- SCADA_OFF_NOMINALOff-nominal SCADA Pseudo Alarms. Non-product and not included with the ALL option

-allowReset

Allow a full model reset to be performed, for testing and/or training. This should NEVER be specified on a production system.

-alphaZoneSort

This option specifies whether DDService should sort zones alphabetically by zone name before sending them to the authority tool for display.

Without this option, DDService sorts zones by the CONTROL_ZONES.sequence number. This column is populated implicitly by the nms-parse-zones script, and follows the ordering in the control_zones.dat file. Projects can choose to reset the sequence column to change the order of the zones if the so choose.

-autoclear <string>

Activate automatic clearing of DDS alarms and purging of FLM and SCADA history. If this option is used with no other arguments then DDS alarms will be cleared hourly from the time DDService is initialized. If you supply an argument in the form "#:#:A" The first number is the amount in days. The second number is the time of day (0-23, 0 being 12:00a). The last item in the argument should either be an "A" or nothing. If you give it invalid arguments, then it reverts to clearing all DDS alarms daily (1:23:A).

example: 0:23

which will cause all DDSalarms to be cleared at 11:00p

example: 1:23

which will cause DDSalarms older than one (1) day to be cleared at 11:00p

example: 40:23

which will cause DDSalarms older than forty (40) days to be cleared at 11:00p

and so on.

If an **A** is last in the argument, all alarms will be cleared.

If an **A** is not supplied, only acknowledge alarms will be cleared.

Historical FLM and SCADA rows will be purged at the same time, as configured by the "RETAIN_HISTORY_RECORDS" ces_parameters entry. This defaults to purging rows that are over 4 weeks old.

-czTimeOut <number>

Set the time out as the number of seconds for control zone transferring. If one user sends out a request and the other user doesn't response in the number of seconds, it will be time out. The default number is 300 seconds or 5 minutes.

Note: The option is only useful with -zones and without -subscribezone.

-dbname <name>

Connect to the DBService started with the `-service <name>` option. For example, if you started:

```
DBService -service dd -process_name DDDBService
```

Then you would specify:

```
DDService -dbname dd
```

to connect to it.

-doNotSyncSCADA

Do not automatically remove the Instruct conditions when a requested SCADA control response is received. This is not recommended behavior.

-delegate <ncg|crew>

Allows DDService to process delegated zones. Without any parameters, defaults to exclusive owner mode in each delegated zone, where only the user who created the zone may operate devices in it.

With -delegate ncg, allows all users with authority over the NCG to operate interior devices. With -delegate crew, prevents all users from operating devices in the zone while a crew is assigned. Once a crew has been released, authority falls to an authorized operator, either the one specific owner (if simply -delegate crew is specified), or to the group of authorized users (if -delegate ncg crew is used).

-lookAheadIgnoreClasses <note,drawing,etc>

Condition classes to skip when populating the Look Ahead Conditions table.

-maxStudySessions <number>

This option will set the maximum number of study sessions allowed. Since each study session takes MTService and PFService memory to hold the Network solution, this number should be set to a point before the services become too large for the operating system. Defaults to 50 if not specified.

-noScadaEvents

Do not generate alarms for SCADA events. Use this option to avoid duplication when the SCADA interface passes all alarms to DDService.

-override

This option will override the wait period for a telemetered device when it has been opened/closed. The telemetered device is defined as having an entry in the digital_measurements table, but not in the controls table.

When manually operated, the digital measurement of attribute 0 remains the status reported via the SCADA interface.

-preventGroundedTagRemoval

Prevents removal of tags from devices in a grounded (or earthed) segment.

-sendAsyncSCADA

Use this option to enable web service based SCADA interfaces. These have to be asynchronous messages because they cannot subscribe to isis process groups.

-subscribezone

Set the control zone management mode to subscription mode which allows multiple users to own a zone. The default mode without this option is exclusive mode which means any time any zone can only owned by one user.

-summaryobjects

Switch on summary objects - conditions can be then be mapped to more a second device for the purpose of summarizing conditions on objects held on other maps.

-systemNCG <integer>

The NCG to use for system alarms.

-validateOperation

Check that device CLOSE operations will not connect two hot phases.

Note: This operation adds overhead to device operations and is only useful when cross-phase objects are present in the model.

-zones

Enable control zone authorizing in DDService.

JMService

-dbname <name>

Connect to the DBService started with the -service <name> option. For example, if you started:

```
DBService -service tc -process_name TCDBService
```

Then you would specify:

```
JMService -dbname tc
```

to connect to it.

-groupingParam <name1=value1> <name2=value2 > <etc>

Set grouping parameters for the passes name/value pairs. This configuration is uncommon. Available names:

- GRPQ_BLOCKSIZE_HIGH - The number of calls we try to process per grouping iteration when group queue is congested. Default: 350.
- GRPQ_BLOCKSIZE_LOW - The number of calls we try to process per grouping iteration. Default: 200.
- GRPQ_USER_REQ_HIGH -The minimum number of calls processed before allowing user requests when group queue is congested. Default: 25.
- GRPQ_USER_REQ_LOW -The minimum number of calls processed before allowing user requests. Default: 5
- GRPQ_CONGEST_LEVEL -The size of the grouping queue at which point it is considered *congested*. Default: 5000
- GRPQ_MAX_USER_REQ - The number of user_requests before JMService breaks out of the grouping thread immediately. Default: 2.
- GRPQ_WAIT_TIMEOUT -The number of milliseconds the grouping thread will wait for user_requests to get down to GRPQ_MAX_USER_REQ. Default: 120000
- MAX_SRSOUTPUT_BYTES – The maximum message size for broadcasts. Default: 120000.
- STABLE_WAIT_THRESHOLD – The low and high broadcast threshold. Value is in the form of <low number, high number>. Default: 2,3

-noAutoERT

Do not automatically calculate ERTs for jobs. Only allow users and/or interfaces to assign ERTs.

-noBroadcaster

Do not broadcast outages. Only used when interfacing with external OMS interfaces.

-noGrouping

Do not group outages. Only used when interfacing with external OMS interfaces.

MBService

-dbname <name>

Connect to the DBService started with the -service <name> option. For example, if you started:

```
DBService -service mb -process_name MBDBService
```

Then you would specify:

```
MBService -dbname mb
```

to connect to it.

-deleteConditions [inc PO RO <condClass>]

Allow devices with conditions to be deleted and reported. If not set, the build will abort when attempting to delete devices with conditions. Takes a space-separated list of condition names.

-deleteOffNominals

Allow devices which are off-nominal to be deleted and reported. If not set, the build will abort when attempting to delete devices with off-nominals.

-deleteOffNominalCls [<device class> <device class>]

Allow devices of the configured classes which are off-nominal to be deleted and reported. If not set, the build will abort when attempting to delete devices with off-nominals. Takes a space-separated list of device class names.

-destinations

Specify destinations to be used for error logging.

string

A list of desired error logging destinations. Options:

D - database

E - standard error (MBService log file)

F - patch file (in $OPERATIONS_MODELS/errors/)

B - CMMS message broadcast

S - System log (syslog directory)

N - No destinations

The various destinations should be strung together: SED

-disableConnectionCheck

When performing model builds, don't assign new generic nodes when the number of connections to a node changes.

-lookupChangedNCG

Use the feeder_ncg table to identify a new NCG value if an alternate view with a new partition is promoted

-nosrs

Signifies that JMService is not being used in this system. Also used for model build environments, or for initial builds.

-offline

MBService is in an initial model build where all services are not required.

-previewMaps

Generates preliminary maps after a patch is applied.

-report

Generate a report of all changes made for the patch.

If this option is not on the command line, no report will be made. The report will be in readable form and found in file "data/reports/Patch#.report".

-export[-exportxml]

As the Model Build Service (MBS) updates the model and generates updated map files (.mad files), if -export is set, MBService will also generate export .mb files for external systems to pick up the latest model definition. These export files will be placed in the $OPERATIONS_MODELS/export directory and there will be a 1:1 relationship to the MBS generated .mad files.

If -export is set, a second option, -exportxml, is available to include an .xml formatted export file in the same location, $OPERATIONS_MODELS/export. An xml schema file for the .xml export is available in this location $NMS_BASE/product/sql/mb.sql.

-sortFeeders

Auto-populate the SORT_ID attributes of all switches in each feeder-based geographic partition.

That is, starting from the FID object, MBService will assign integers to each downstream switch, and populate the SORT_ID attribute with the switch ID and the parent switch's ID, separated by the '-' sign.

For example, a breaker would be '`00000-00000`,' a downstream recloser would be '`00001-00000`,' and a fuse downstream of the recloser would be '`00002-00001`,' and a transformer downstream of the fuse would be '`00003-00002`,' etc.

The appropriate columns must be in the attribute tables, and the DEVICE_ATTRIBUTES must be configured correctly to save the SORT_ID attribute. The customer must also have feeder-based partitioning.

MTService

-dbname <name>

Connect to the DBService started with the -service <name> option. For example, if you started:

```
DBService -service mt -process_name MTDBService
```

Then you would specify:

```
MTService -dbname mt
```

to connect to it.

-groundThroughFIDs

This flag allows projects to propagate the GROUND color through feeder heads. Note that grounded and energized produces a FAULT color, whose behavior is undefined about feeder heads. Also note that this decreases MTService performance, as it now potentially needs to solve an entire island, instead of a single feeder.

-ignoreloops

Loop processing is a significant expense during topology processing. If loop colors are not necessary, this option is used to disable all loop processing.

-includeDeviceOnDownstream

Add any SND connected to a device to the list of downstream affected customers in Look Ahead.

-phaseCheckXFMs

Accepts a space-delimited list of transformer classes, which MTService will test against the nominal model. If any instance of the listed classes is found to be partially energized, MTService will print a log warning describing the problem, along with the alias into the log file.

MTService will then continue to operate under the assumption that the devices are incorrectly modeled and should only have the hot phases available. This prevents difficult situations within outage management where SNDs can never be reenergized after outages, due to modeling errors.

Note that this only executes on startup, not after model builds. That is, when a map is rebuilt, MTService must assume that it has been corrected, and cannot afford to resolve the nominal network and recheck for disconnected transformers.

This option should be removed as soon as the model has been cleaned, since it impairs MTService startup performance considerably.

-simple_network_protector_solves

Disable complex HV/LV network protector processing.

ODService

-aggregates

Cache aggregate definitions for performance in the Feeder Focus operation.

-cache <classes>

Builds an AliasCache of the specified devices.

-db_types <type type etc>

ALIAS_MAPPING.DB_TYPE values to load.

-delay

Used to allow the system to start up before loading device alias information. For example, if the services take 8 minutes to launch, then a -delay 10 could be used to delay alias caching.

-ignore

Specifies the "db_types" to ignore when caching aliases; db_types include DNO, OPS, DDB, and RTI_SP.

-locations

Cache object locations.

-mru (Most Recently Used)

Creates an array of size (n) to store device aliases. This helps when the same alias is looked up repeatedly. Note: The process for determining an alias is to first check the MRU cache, then check the general cache, then finally execute a DBService query if an alias has not been found

-noAliases <class class etc>

Do not query aliases for the specified class names.

-period <n>

Seconds for initialization retry (minimum 5 seconds).

-stats

Enable debugging of mru cache statistics.

High Level Messages

High level messages can be sent to the NMS services using the syntax:

```
Action any.<service name> <action command>
```

The service name component can include a wildcard (*) to match multiple services.

This section contains the following High Level Message categories:

- Action Commands Supported By All Services
- DDService
- JMService
- MBService
- MTService
- ODService

Action Commands Supported By All Services

DEBUG

Toggle debugging to 1 or 0 for all facilities.

DEBUG <integer>

Set all debug levels to the specified level.

DEBUG <name> <integer>

Set the specified facility to the level.

DISPLAYMESSAGE <message>

Prints the message to the log file. This is helpful when debugging, to help determine what actions the user is taking and when.

FUNCTIONTRACE

Turn on function trace debug.

ISISDUMP

Triggers an isis dump.

PGTIMING ON

Turn on process group timing debug.

RELOG

Generates a new log file.

REPORT

Returns the number of matching processes.

STOP

Stops the service. Note that this can only stop a subset of services.

The `Action any.any STOP` command has been replaced by the `sms-stop` command. If you require its use, add the `-force` option:

```
Action -force any.any STOP
```

DDService

CHK_ABNORMAL

Audit the Abnormal Device Summary, and add or remove records as needed.

DELETE_CONDITION <Handle>

Deletes the passed condition from the Real-Time session.

DUMP <parameter>

Dumps the given structure. If none provided, dumps all of them.

- **analogs:** analog measurements
- **conditions:** all conditions
- **controls:** outbound controls
- **ctrl_zone:** ctrl_zone_mgr
- **dcz:** delegated_zone_mgr
- **digitals:** digital measurements
- **event:** event_mgr
- **measures:** same as analogs & digitals
- **momentary:** pending momentaries
- **rt_switch:** switch_mgr
- **rules:** srs rules
- **session:** session_mgr
- **<handle>:** dump digital/analog info for handle <cls.idx.att>

PURGE_HISTORY

Runs the customizable PURGE_HISTORY_TABLES() database procedure to purge records older than the CES_PARAMETERS 'RETAIN_HISTORY_RECORDS' configured number of weeks from the following tables:

- PF_DERMS_FORECAST_HISTORY
- FLM_FDR_LOAD_HISTORY
- FLM_FDR_LOAD_DETAILS_HISTORY
- FLM_DEV_VIOLATIONS_HISTORY
- SCADA_ANALOG_HISTORY

RELOAD_RULES

Reload the SRS_RULES from the database.

WHAT <handle or alias>

Print the real-time information for the passed device handle.

JMService

CLEAR_OLD_CALLBACKS <timeout>

Clear callbacks for all events for which time passed since restoration exceeded value specified by the message argument. Takes the callback expiration timeout in minutes.

DUMP <parameter> <parameter>

Print current internal status of JMService. If no arguments are used, defaults to "rules queues feeders completed crews upstream_cache statistics allcalls."

One or more parameters may be specified:

- **active:** Output information about active events from res_by_event hash. Inactive events always printed.
- **alarmqueue:** Output content of the timed alarm queue.
- **alarms:** Output content of the res_by_alarm hash table.
- **allcalls:** When used in conjunction with the "feeders" option, prints all calls on each trouble feeder.
- **amr:** Dump AMR state.
- **autoert:** Print configuration used to assign initial ERTs to events.
- **avl:** Dump AVL state.
- **caches:** Print the contents of miscellaneous JMSService caches not included in other options. Currently, just the cache of disconnected customers.
- **cb:** Dump Callbacks state.
- **completed:** Print all completed events held in memory.
- **crews:** Print information about crews.
- **events:** Print all of the active and inactive events in memory.
- **feeders:** Print all of the trouble feeders.
- **jobs_by_device:** Print contents of jobs by device cache.
- **jobs_by_sheet:** Print contents of jobs by switch sheet cache.
- **memstats:** Print statistics about JMService memory usage.
- **queues:** Print all of the queues in JMSService. This includes the grouping queue (calls waiting to be grouped).
- **rules:** Prints information about JMService rule sets and rules. A rule set must be active in at least one control zone before it will appear in the output.
- **statecb:** Print TE state callbacks.
- **statemgr:** Print TE state manager.
- **statistics:** Print hash table lookup statistics, statistics about calls processed, and other performance statistics.
- **stormman:** Prints state of the Storm Management module (large amount of output).
- **supplynodes:** Print supply node caches.
- **upstream_cache:** Print cache of upstream devices.
- **<Class.Index>:** If passed an event handle, prints the specified event. If passed a trouble feeder handle, prints all events on that trouble feeder. If passed a device

handle, prints all events on that device. If passed a call condition handle (class of 801), print the specified call. Special trouble feeder handles can be passed as follows:

`911.911`: The no processing trouble feeder.

`7.7`**:** The non-outage trouble feeder.

`7.0`: the fuzzy trouble feeder.

`810.810`: the momentary trouble feeder.

`1.1`: the callback trouble feeder.

- **<DeviceAlias>:** If the parameter matches an alias in ODService, prints information for that device (or trouble feeder) handle as specified above.

FIX DUPLICATES <event handle>

This is a non-standard message used in the unlikely event you have duplicate event numbers in the JOBS table and/or the Work Agenda for non-NFY events. Consult Customer Support if you are thinking about using this message.

GROUPING_PARAM <name1=value1> <name2=value2 >

Set grouping parameters for the passes name/value pairs. This configuration is uncommon.

Available names:

- GRPQ_BLOCKSIZE_HIGH: The number of calls we try to process per grouping iteration when group queue is congested. Default: 350.
- GRPQ_BLOCKSIZE_LOW: The number of calls we try to process per grouping iteration. Default: 200.
- GRPQ_USER_REQ_HIGH: The minimum number of calls processed before allowing user requests when group queue is congested. Default: 25.
- GRPQ_USER_REQ_LOW: The minimum number of calls processed before allowing user requests. Default: 5
- GRPQ_CONGEST_LEVEL: The size of the grouping queue at which point it is considered "congested." Default: 5000
- GRPQ_MAX_USER_REQ: The number of user_requests before JMService breaks out of the grouping thread immediately. Default: 2.
- GRPQ_WAIT_TIMEOUT: The number of milliseconds the grouping thread will wait for user_requests to get down to GRPQ_MAX_USER_REQ. Default: 120000
- MAX_SRSOUTPUT_BYTES: The maximum message size for broadcasts. Default: 120000.
- STABLE_WAIT_THRESHOLD: The low and high broadcast threshold. Value is in the form of <low number, high number>. Default: 2,3

ISIS_PRM <filename>

Change or report the current isis parameter file. If specified, this is the new isis paramter file that will be used. Otherwise, the current parameter file in use will be output the service log.

RELOAD <options> <parameters>

Reload the specified data structures.

Options:

-wait

If specified, JMService will wait until the reload actions are all finished before sending a reply message. Otherwise the reply is sent before doing the first reload operation.

Parameters:

Any number of these parameters may be specified.

- **crew_members:** reload crew_members table.
- **crew_vehicles:** reload crew_vehicles table.
- **statecb:** reload te_state_callbacks and te_state_cb_args tables.
- **dispatch_groups:** reload dispatch group (referral group) config tables.
- **stormman:** reload Storm Management configuration.
- **trouble_codes:** reload trouble code configuration from srs_trouble_codes and call_quality tables.

RELOAD_RULES

Reload all JMService configuration rules.

SIMULATION <start time>

Set/unset a fixed Storm Management simulation start time.

No argument will unset any fixed time and revert back to current time. An argument consisting of a datetime string in double quotes (*e.g.*, "01/02/03 04:05") will cause the simulation start time to be fixed at that date and time.

If the "simulation" option is provided an argument, this argument is assumed to be a datetime string (in a format specified by the NMS_SYSDATE environment variable) and converted to a time_t value. If only a date or only a time is provided as an argument, then the rules of the getdate() system command apply with regard to how the argument is converted into a full datetime string. If the argument cannot be successfully converted, then the command behaves as if no argument was provided at all. If multiple arguments are provided, only the first one is used -- the rest are ignored (thus, an argument consisting of a date and time must be enclosed in double quotes in order to be processed correctly). If no argument is provided, the command will unset any currently applied fixed simulation time and return to using the current time for any subsequent simulations. If either the -stormman command-line option is not in use or the Storm Management functionality failed to properly initialize upon JMService startup, then the "simulation" command is effectively a no-op.

VALIDATE <parameter>

Validates the JMService state. Errors and warnings are output to the service log.

One of the following parameters must be used.

- **jobs:** Validates the integrity of JMService memory structures holding jobs
- **jobs_in_db:** Validates a one-to-one relationship between jobs in JMService memory and jobs in the jobs database table.
- **snodes:** Validates the integrity of JMService memory structures holding supply nodes
- **snodes_in_db:** Validates a one-to-one relationship between supply nodes in JMService memory and supply nodes in the supply_node_log database table.
- **calls:** Validates the integrity of JMService memory structures holding calls

- **calls_in_db:** Validates a one-to-one relationship between calls in JMService memory and calls in the incidents database table.
- **all:** Validates all of the above.

MBService

HALTBUILD

Suspend model builds after any current patches finish.

RESUMEBUILD

Resume model build processing.

MTService

COLORS

Request MTService to reload feeder colors.

DUMP MEMSTATS

Dump memory statistics to the log.

DUMP_COLORS

This will show debug information on all the colors stored by MTS.

DUMP_NETWORK

Dump the entire electrical network for debugging.

STUDY_WHAT <session> <handle>

Print the study mode information for the passed device handle.

WHAT <handle or alias>

Print the real-time information for the passed device handle.

ODService

DUMP_CACHE

Dump the cached information for debugging.

METRICS

Dump cache utilization metrics.

Appendix B

Model Edit Configuration

The ControlEdit program must always be given a basic edit function. At this time, there are four main options available, as well as removal options:

Parameter	Description
-inlineCross	Inserts an inline cross phase jumper into the model at the head of the selected spur. This is used to correct mis-phased spurs. You must also specify -crossPhase option (see Other Options on page B-2). Configure the JBot Command as InlineCrossJumperCommand. This is rare.
-jumper	Inserts a jumper between the selected conductor and another conductor to be chosen interactively by the user. If this is to be a cross phase jumper, you also need to specify the -crossPhase option (see Other Options on page B-2). Configure the JBot Command as PPJumperCommand or PPCrossJumperCommand.
-tap	Inserts objects as taps to the selected conductor, such as generators. Configure the JBot Command as TapGeneratorCommand.
-wireDown	Breaks a conductor and inserts an inline switch. Usually configured to call the InlineJumperCommand, but, when used with the BypassJumperCommand, also includes a bypass switch. See -bypassCls, -bypassSymbolCls, -bypassDistance in Other Options on page B-2.
-remove	Removes the model edit. This can be used for any model edit type, and renders -putUpWire and -removeJumper obsolete. Configure the JBot Command as RemoveJumperCommand.
-putUpWire/-removeJumper	Removes a device installed previously by ControlEdit on a wire down or jumper edit. Configure the JBot Command as RemoveJumperCommand.

Other Options

Note that all phase-based options are automatically filtered by the phases of the selected device, so you can configure 7 (ABC) for single-phase edits.

Option	Arguments	Description	Req
-alias	String	The alias to use for the new switch. Also see "Prompt Dialog Configuration" below.	N
-attribute	String, the <name:value>	The name and value pair to use for the attribute name. Separate these by a colon. You may specify –attribute as many times as you wish, to set many different attributes. Also see "Prompt Dialog Configuration" below.	N
-bypassCls	String	Class to use as a bypass switch for the inline jumper. **Note:** Required when using BypassJumperCommand.	N
-bypassDistance	Integer	Distance from the inline jumper to the bypass switch. Used with the BypassJumperCommand, but not required. Defaults to 25.	N
-bypassSymbolCls	String	Symbology class used for the bypass switch. **Note:** Required when using BypassJumperCommand.	N
-cesuser	String, or $CESUSER	The user name	Y
-connection/-c	Integer, generally 7 or $PHASES	The available phases. Defaults to 7 (ABC)	N
-coordSys	Integer, or $COORDSYS	The coordinate system number. Defaults to 0 (geographic).	N
-crossPhase/-cpj	<none>	Whether to perform a cross-phase jumper. Used with the –jumper option. This asks the user to choose another conductor, then displays a live/pickup phase chooser dialog to the user.	N
-gapLength	Integer	The gap to use for the switch being added. Defaults to 25.	N
-handle	The Handle, or $DEVICE	Handle of object clicked	Y
-inlineObjects/-io	See Inline Objects Configuration below	A variable-length list of all classes to add, along with configuration for each.	N
-noVerify	<none>	Whether to skip the control zone verification. Not recommended.	N
-symbolCls	Integer	The symbol class to use for the new switch. Defaults to the switch class.	N

Option	Arguments	Description	Req
-useStdCls	<none>	Whether to use default classes (ces_ap_cond, ces_bp_cond, ces_cp_cond, ces_2p_cond, ces_3p_cond) for the new conductors. If not specified, it uses the selected conductor class. Not recommended.	N
-x	The x-coordinate, or $WX	X coordinate where action takes place	Y
-y	The y-coordinate, or $WY	Y coordinate where action takes place	Y

Substitution Parameters

Substitution parameters can be added as arguments to any of the above options. For example, you can specify "-symbolCls 4460$PHASES", which will use symbol class 44601 for A-phase devices, 44607 for ABC-phase devices, etc.

Name	Value
$CESUSER	The user who initiated the edit
$COORDSYS	The coordinate system of the selected device, as defined in the map.
$DEVICE	The selected device Handle.
$PHASES	The selected device phases, 1-7
$WX	The x-coordinate
$WY	The y-coordinate
$XPHASE	Combined crossed phases, for use in the -symbolCls option, if desired. For example, an A->B cross phase jumper will contain 3 (AB), and an A->C jumper will contain 5 (AC), etc. This is only valid for cross-phase model edits.

Prompt Dialog Configuration

You can configure these model edit options to prompt the user to enter values used in aliases and other attributes. These are then specified as arguments to the above options, similarly to the Substitution Parameters above.

Enter the full set of options in the **CONTROL_ACT_PROMPTS** table:

Column Name	Description
prompt_key	The primary key of this table. May not be NULL or contain spaces.
prompt_string	The String to present to the user to describe the value to enter
required	Y or N. If Y, this value must be entered and non-blank. If N, this value can be left blank
data_type	<string, integer, number, list, fixed_list> This is not enforced in the initial implementation and will be used in a future release. At this time, all prompts will assume string values.
default_value	Default value for the prompt and auto-populated in the prompt dialog.
list_values	List of values to populate in a combo-box for the user to select the value to use. Values will be separated by "|" delimiters. ie .Emergency|GIS Data Error|Planned|Other. This is not supported in the initial implementation and will be used in a future release.

If powerflow attribution is required, there should be a CONTROL_ACT_PROMPTS record for each attribute column in the PF_DIST_GEN_DATA database table. This will then use configured values in the PF_DIST_GEN_DATA table for each catalog_id record where model_edit = 'Y'.

Then specify these in your CONTROL_ACT description column to prompt the user to enter attributes. For example:

Configure a default jumper alias:

```
INSERT INTO control_act_prompts (prompt_key, prompt_string, required,
data_type, default_value, list_values)
VALUES ('JmpAlias','Jumper Alias','Y','string','Jumper_$WX_$WY','');
```

Then configure a jumper action that prompts the user to enter an alias. The dialog will be prepopulated with "Jumper_<x-coordinate>_<y-coordinate>":

```
INSERT INTO control_act
  (act_key, act_cls, act_idx, action_name, label, instruct_label,
   switching_desc, switching_code, description, undo_act_key)
VALUES
  (481, 'START', 'ControlEdit', 'ACT_PP', 'P-P Jumper Prompt...', '',
   '', '', '-jumper -deviceCls 447 -status 0 -nominal 0 -gapLength 50 -
noVerify -handle $DEVICE -cesuser $CESUSER -x $WX -y $WY -alias
$PROMPT(JmpAlias) -symbolCls 4470$PHASES -coordSys $COORDSYS', 0);
```

Also configure the CONTROL_ACT_DATE_FORMATS table. This table specifies the date formats for string substitution.

Column Name	Description
format_key	The primary key of this table
format_string	The format string to be used, based on Java SimpleDateFormat

For example:

Enter a default date format.

```
INSERT INTO control_act_date_formats (format_key, format_string)
VALUES ('DEFAULT','yyyy-MM-dd_HH:mm:ss');
```

Substitute it into the alias

```
INSERT INTO control_act
  (act_key, act_cls, act_idx, action_name, label, instruct_label,
   switching_desc, switching_code, description, undo_act_key)
VALUES
  (480, 'START', 'ControlEdit', 'ACT_PP', 'P-P Jumper', '',
   '', '', '-jumper -deviceCls 447 -status 0 -nominal 0 -gapLength 50 -
noVerify -handle $DEVICE -cesuser $CESUSER -x $WX -y $WY -alias
JMP_$DATE(DEFAULT)_$CESUSER -symbolCls 4470$PHASES -coordSys
$COORDSYS', 0);
```

Inline Objects Configuration

The –inlineObjects/-io option allows you to specify a variable-length list of objects, in the desired order. Each inline object definition is separated by a colon:

```
-inlineObjects <inline obj def> : <inline obj def> : <inline obj def> …
```

And each <inline obj def> consists of the following:

1. <class num or name>
2. <symbol number>
3. <nominal status>
4. <current status>
5. <branch length>
6. (optional) ALIAS:<alias value>
7. (optional) ATTRIBUTE:<attribute name>:<attribute value>

ATTRIBUTE can be repeated as many times as desired.

There are two reserved keywords that can be used in place of the <symbol number> and <branch length>:

- DEF - use default value
- COND - create a handle based on the class of the parent conductor

For example, the "'Tap Generator with switch Prompt" option:

```
INSERT INTO control_act
  (act_key, act_cls, act_idx, action_name, label, instruct_label,
   switching_desc, switching_code, description, undo_act_key)
VALUES
  (476, 'START', 'ControlEdit', 'ACT_TAPGEN', 'Tap Generator with
switch Prompt...', '',
   '', '', '-tap -noVerify -gapLength 40 -x $WX -y $WY  -handle $DEVICE
-io COND DEF 7 7 DEF : 148 14807 0 0 DEF ALIAS:$PROMPT(SwAlias)
ATTRIBUTE:gmd_comment:$PROMPT(SwComment)
ATTRIBUTE:gmd_comment2:$PROMPT(SwComment2) : 191 19107 7 7 DEF
ALIAS:$PROMPT(GenAliasFdrName) ATTRIBUTE:comment_1:$PROMPT(GenComment)
: 998 0 7 7 DEF ATTRIBUTE:FEEDER_NAME:$PROMPT(GenAliasFdrName)
ATTRIBUTE:SUBSTATION_NAME:$PROMPT(GenStationName)
ATTRIBUTE:COLOR:$PROMPT(FdrColor) : 448 0 7 7 DEF : 996 0 7 7 DEF : 448
0 7 7 DEF : 299 0 7 7 DEF -status 7 -nominal 7 -deviceCls 148 -
symbolCls 14807 -fdrColor 6 -cesuser $CESUSER -coordSys $COORDSYS',
0);
```

The io option is "COND DEF 7 7 DEF : 148 14807 0 0 DEF ALIAS:$PROMPT(SwAlias) ATTRIBUTE:gmd_comment:$PROMPT(SwComment) ATTRIBUTE:gmd_comment2:$PROMPT(SwComment2) : 191 19107 7 7 DEF ALIAS:$PROMPT(GenAliasFdrName) ATTRIBUTE:comment_1:$PROMPT(GenComment) : 998 0 7 7 DEF ATTRIBUTE:FEEDER_NAME:$PROMPT(GenAliasFdrName) ATTRIBUTE:SUBSTATION_NAME:$PROMPT(GenStationName) ATTRIBUTE:COLOR:$PROMPT(FdrColor) : 448 0 7 7 DEF : 996 0 7 7 DEF : 448 0 7 7 DEF : 299 0 7 7 DEF"

Which creates:

1. A conductor with the selected conductor class, the default symbol, with nominal & current status 7, and the default length based on the gap length.
2. A switch of class 148 with symbol class 14807, nominally open. The user will be prompted for the alias and 2 comments for this switch.
3. A generator of class 191, symbol class 19107, nominally closed. The user will be prompted for the feeder name and comment for this generator
4. An FID (class 998) with no symbology and a prompted feeder name, substation, and color
5. An invisible 448 connector
6. An FBD (996)
7. Another invisible connector
8. A SRC (299)