

**Installation and Customization Guide**

**PowerEditor 5.10**

Last Revised: September 7, 2018

Notice: This document includes confidential and proprietary data that is within the scope of non-disclosure agreements between MindBox and its alliances. The document shall not be duplicated, used or disclosed - in whole or in part - for any purpose other than that specified within the scope of such an agreement. Acceptance of this document presumes adherence to the above conditions.

The PowerEditor documentation suite includes the following materials:

PowerEditor Business Analysts Guide

PowerEditor Custom Reports Guide

PowerEditor Installation and Customization Guide

PowerEditor Reference Manual

PowerEditor Release Notes

PowerEditor Rule Writers Toolkit

PowerEditor Web Service API Guide

*Release Notes* for earlier versions are also provided in the release package. For technical assistance with upgrading or any other PowerEditor-related issue please contact CoreLogic Technical Support at 1-855.369.2410, or [ADESupport@corelogic.com](mailto:ADESupport@corelogic.com)

**Table of Contents**

1. Introduction 6

2. PowerEditor Server System Requirements 7

2.1 OS Requirements 7

2.2 Java Configuration Requirements 7

2.3 Web Application Container Requirements 8

2.4 Miscellaneous Configuration Requirements 9

2.4.1 LDAP Requirements 9

2.4.2 Crystal Reporting Requirements 10

3. PowerEditor Server Installation 11

3.1 Server Installation 11

3.1.1 Server Installation Step 1: Obtain PowerEditor Distribution file 11

3.1.2 Server Installation Step 2: Configure Database and Password. 11

3.1.3 Server Installation Step 3: Modify Server Configuration Options 11

3.1.4 Server Installation Step 4: Deploy PowerEditor WAR File 12

3.2 Server Installation – Database Configuration 13

3.2.1 Microsoft Access DB Configuration 13

3.2.2 MySQL DB Configuration 15

3.2.3 Oracle DB Configuration 15

3.2.4 MS SQLServer DB Configuration 16

3.2.5 Sybase DB Configuration 16

3.2.6 Table Indexing 17

3.3 Password Encryption 17

3.4 Server Configuration Validation and Troubleshooting 18

3.4.1 Trouble-Shooting 19

4. PowerEditor Server Upgrade Procedures 21

5. PowerEditor Client Installation 22

5.1 PE Client System Requirements 22

5.2 PE Client Installation Instructions 22

5.2.1 Basic Instructions 22

5.2.2 Client Installation Refinements 23

5.2.3 PE Client Installation Trouble-Shooting Suggestions 24

6. PowerEditor Customization Guide 28

6.1 Path to PowerEditor Configuration File 28

6.2 KB Filtering With Date Range 28

6.2.1 Configuration of KB Date Filter 28

6.2.2 Sample Set up 29

6.3 High-Level Tab Configuration 29

6.4 Entity Configuration 31

6.4.1 Entity Configuration Overview 31

6.4.2 Entity Definition 33

6.5 User Interface Customization 37

6.5.1 User Interface: Entity Display 37

6.5.2 User Interface: Usage Type List 38

6.5.3 User Interface: Guideline Management 38

6.5.4 User Interface: Multiple Date Synonyms With Same Date 41

6.5.5 User Interface: Miscellaneous Options 41

6.5.6 User Interface: Policy Management 42

6.5.7 User Interface: Show User’s Name in the Title Window 43

6.5.8 User Inteface: Enable/disable User Account Functionality 43

6.5.9 User Inteface: Access Disclaimer on Login Screen 43

6.5.10 Failed Login Attempts Tracking 43

6.6 Rule Generation Options 44

6.6.1 Customize Number of Threads Used for Rule Generation 44

6.6.2 Generate One Rule File Per Usage Type 44

6.6.3 Template Precision Field Use 44

6.6.4 LHS Options 44

6.6.5 RHS Options 48

6.6.6 Message Generation Options 48

6.6.7 Parameter Object Generation Options 49

6.7 Server Customization 49

6.7.1 Knowledge Base 49

6.7.2 Audit Trail 50

6.7.3 Database Connection 50

6.7.4 Deployment 50

6.7.5 Log Files 52

6.7.6 Session Configuration 53

6.7.7 Single-Sign-On Support 53

6.8 Enumeration Source Configuration 54

6.8.1 Enumeration Source Declaration 54

6.8.2 Format of Enumeration XML Files 55

6.8.3 Sample Enumeration XML File 55

6.8.4 Example of Use 56

6.9 LDAP, Authentication, and Authorization 56

6.9.1 Overview and Definitions 56

6.9.2 PE Authentication Configuration Overview 58

6.9.3 LDAP Overview 59

6.10 Password Policies 62

6.10.1 Overview 62

6.10.2 Example Password Policies 63

6.10.3 Lockout Behavior 63

6.10.4 Change Password Restriction 65

6.10.5 Password Policy Configuration File Specifications 65

6.11 Summary of Configuration Changes for Backward Compatibility 66

7. Advanced Topics 67

7.1 Support for Http-Only Cookie 67

7.2 Notify SSO Authentication on Successful SSO Launch 67

7.3 User Tracking Table Support 67

8. Configuration File Reference 68

8.1 Summary of Knowledge Base Filter Parameters 68

8.2 Summary of Feature Configuration Parameters 68

8.3 Summary of Entity Configuration Parameters 69

8.4 Summary of UI Configuration Parameters 70

8.5 Summary of Rule Generation Parameters 71

8.5.1 Rule Generation: Overall Configuration 71

8.5.2 Rule Generation: Guideline Generation Configuration 72

8.5.3 Rule Generation: Object Generation Configuration 74

8.6 Summary of Server Parameters 75

9. Appendix I:LDAP Server - Sample Installation 77

9.1 Obtain Configuration Files 77

9.2 Install and Configure LDAP Server Application (OpenLDAP) 77

9.3 Install LDAP Client Application (JXplorer) 79

9.4 Use Client Application to Configure PowerEditor LDAP Settings 80

9.5 Restarting JXplorer 84

9.6 Debugging Open LDAP 84

# Introduction

The following document provides technical specifications for installing and configuring the MindBox PowerEditor. This document should be used in conjunction with the *PowerEditor Release* *Notes* that are specific to the release you are using. For an overview of the PowerEditor from a user’s perspective, please refer to the *PowerEditor User’s Guide*, which is a document distributed with each software release.

This document provides the following information:

* System Requirements
* General Installation Instructions
* Configuration Instructions

For technical support, contact MindBox Technical Support at [Support@mindbox.com](mailto:Support@mindbox.com), or 1-888-650-6463.

Other documents distributed with the PowerEditor include:

|  |  |
| --- | --- |
| *PowerEditor Business Analysts Guide* | Intended for Business Analysts who are using the PowerEditor to develop business rules. |
| *PowerEditor Custom Reports Guide* | Describes the report-writing API provided by the PowerEditor. It provides examples for writing Crystal Reports that contain PowerEditor data. |
| *PowerEditor Reference Manual* | Provides an overview of PowerEditor and describes various features, considered as general users’ guide. |
| *PowerEditor Rule Writers Toolkit* | Intended for technically savvy users, provides description of functionality that are not covered in the Business Analysts Guide or the Reference Manual. |

# PowerEditor Server System Requirements

The PowerEditor (PE) is a web-based client/server application. This section describes the system requirements for the machine that hosts the PowerEditor server application. See *Section 5 PowerEditor Client Installation* for system requirements for the PowerEditor client.

The PowerEditor server is distributed as a Java web application, and therefore can be run on any operating system that runs Java 1.6 or later. The machine that runs the PowerEditor server machine must be configured with the following components:

* Java SDK with JCE unlimited strength jurisdiction policy
* A Java application server
* Connection to relational database that will contain PowerEditor data

Specific requirements for these components are described in the sections to follow.

## OS Requirements

The PowerEditor server software can be run on any operating system (OS) that runs Java 1.6 or later, with JCE unlimited strength jurisdiction policy.

##### Windows

The PE server software can run on any Windows NT-based OS. This includes, but is not limited to the following Windows operating systems:

* Windows 2000
* Windows XP
* Windows 2003 Server

PowerEditor can also be run in a Linux, Mac OS X, or Unix environment.

Development server machines should minimally be configured with the following:

* 1 GB Ram
* 10 GB Disk Space

Production server machines should be minimally configured with the following:

* 2 GB Ram
* 20 GB Disk Space

##### VMware®

The PE server has also been tested with VMware®. The VMware memory requirements are at least 512MB for development and testing servers, and 1GB for production servers.

*Note: Although the PowerEditor server and client software can be run on a variety of operating systems, this document contains examples from Windows XP.*

## Java Configuration Requirements

The PowerEditor server machine must be configured with the Sun JDK (Java Development Kit) 1.6 or later. The JDK is sometimes referred to as the J2SE SDK (Java 2 Standard Edition, Software Development Kit). Note: Do not confuse the JDK with the JRE. The JRE (Java Runtime Environment) is not sufficient for the PowerEditor *server* machine, although it is sufficient for the PowerEditor *client* machines.

The Sun JDK is available at <http://www.oracle.com/technetwork/java/javase/downloads/index.html>. Again, be sure to download the JDK and not the JRE. Note that the download file is quite large. Once downloaded, execute the file and follow the installation instructions on the screen.

**IMPORTANT Java Compatibility Note**

PowerEditor versions 5.9 and above requires Java version 1.6.

Ensure that Java Cryptography Extension (JCE) unlimited strength jurisdiction policy files are installed. The JCE policy files for Java 1.6 can be found at <http://www.oracle.com/technetwork/java/javase/downloads/jce-6-download-429243.html>.

## Web Application Container Requirements

The PowerEditor server machine must be configured with a Java *application container*, which a mechanism for extending the functionality of a Web server for Java servlets. This can also be referred to as a servlet container or more loosely as a web application server. Examples of Java servlet containers include Apache Tomcat, JBoss Application Server, or Resin. You can find out more about Java servlet containers via:

<http://java.sun.com/products/servlet/overview.html>

The application container must be “War” file compliant and must support Servlet 2.5 and JSP 2.1.

PowerEditor must be run on a single instance of a servlet container. Running a cluster of servlet container instances that each serves the PowerEditor will lead to cache synchronization issues, which can lead to database corruption.

##### Memory Requirements

The application container has the following memory requirements for running the PE Server:

* Development machines should be allocated at least 1GB of memory
* Production machines should be allocated at least 2GB of memory

The procedure for configuring memory allocation is dependent upon the vendor-specific application container – see vendor documentation for memory configuration options. Typically, this is achieved with specifying ‘-Xmx’ java runtime option.

##### Servlet Start-Up

On production machines running in Windows, it is recommended that your application container be installed as a Windows Service. In this way, the PowerEditor server remains will automatically be launched when the server machine is booted. This can be achieved in the Windows Service control panel. The procedure for installing a application container as vendor-specific. Consult vendor documentation for details.

##### Servlet Port Number

You’ll need to know what port number your application container uses for application requests, and make sure that no other applications on your server machine are using that port. For example, Apache Tomcat uses port 8080 by default, which is configurable in the service.xml file in the conf directory. Other application containers specify this differently.

##### Servlet Container Log Files

You’ll need to know where your application container stores its log files, so that you can troubleshoot potential configuration issues. This log file is not to be confused with the log files that the PowerEditor creates.

##### Sample Servlet Containers

The following presents various Java application container options that have been tested with the PowerEditor.

Tomcat The PowerEditor requires Tomcat 6.0 or above. Tomcat can be downloaded via:

<http://tomcat.apache.org>.

To use the PowerEditor 5.9 and above with Tomcat 7.0, ensure the useHttpOnly attribute of <Context>element is set to false in the *conf/context.xml*. See <http://tomcat.apache.org/tomcat-7.0-doc/config/context.html> for details.

Resin The PowerEditor requires Resin EE, Version 2.1 or above. Resin is available at

<http://www.caucho.com/download/>.

Once the download is unzipped, Resin can be launched by running the following executable:

*<resin-dir>*/bin/httpd.exe

JBoss AS JBoss Application Server can be downloaded via [http://jboss.org](http://labs.jboss.com/portal/jbossas)

WebLogic Information on BEA’s WebLogic server is available via [http://www.bea.com](http://www.bea.com/).

WebSphere Trial version of IBM’s WebSphere is available at

<http://www-128.ibm.com/developerworks/websphere/downloads/>

##### Database Requirements for PE Server

PowerEditor supports the following databases:

* MySQL
* SQL Server
* MS Access
* Oracle
* Sybase

The following lists specific DB versions that the PowerEditor has been tested with. If you wish to use versions other than those listed here, please consult with MindBox technical support.

* MySQL 5.6.13 for Windows
* SQL Server 2008 – the page size should be configured to be 8k rather than default of 2k
* Microsoft Access 2003
* Oracle 10.0.1
* Sybase 12.5.x

## Miscellaneous Configuration Requirements

### LDAP Requirements

The PowerEditor can optionally be configured for user authentication and authorization using the LDAP API. See *Section 7.7 LDAP, Authentication, and Authorization* for more details.

PowerEditor supports LDAP v3.

### Crystal Reporting Requirements

The PowerEditor provides for generating reports using Crystal Reports. See the PowerEditor Custom Reports Guide for more details.

Crystal Reports XI Release 2 Developer Edition is required for building PowerEditor reports. PowerEditor Version 5.9 has been tested with Crystal Reports Version 11.0.0.2002.

# PowerEditor Server Installation

The following section assumes that server machine that will be running the PowerEditor server software has been configured as per the system requirements outlined above in *Section 2 PowerEditor Server System Requirements.* In summary, this section assumes that the following have been installed on the server machine:

* OS configured with correct memory requirements
* Java JDK
* Java Application Container
* Database Software
* Database Connector for your Database Software (e.g. ODBC or JDBC)

This chapter contains 3 major sections:

* Server Installation
* Database Configuration
* Password Encryption
* Installation Validation and Troubleshooting

## Server Installation

Installing the PowerEditor server is comprised of the following steps:

1. Obtain distribution file and prepare installation directory
2. Configure an empty PE database, its database connector and its password
3. Modify Server Configuration Options
4. Deploy PowerEditor WAR File

### Server Installation Step 1: Obtain PowerEditor Distribution file

1. Obtain PowerEditor distribution zip file. Clients can get the distribution zip file from <http://softdist.mindbox.com>. MindBox employees can obtain this from <https://mindshare.mindbox.com/FileLib/Public/Editors>.
2. Create a new directory for installation: e.g., c:\mindbox\powereditor. We shall refer to this as <install-dir>
3. Unzip the distribution zip file into the installation directory. You should see a sub directory named ‘config,’ ‘database,’ ‘docs,’ ‘ldap,’ ‘tool,’ and ‘webapp.’ We will use files from these sub directories.
4. Create a sub directory named ‘generated-files,’ and ‘log.’

### Server Installation Step 2: Configure Database and Password.

See *Section 3.2* Server Installation – Database Configuration and *Section 3.3 Password Encryption.*

### Server Installation Step 3: Modify Server Configuration Options

PowerEditor uses XML configuration files. They are found in the <install-dir>/config directory. The following table lists the configuration files of the PowerEditor.

|  |  |
| --- | --- |
| Filename | Description |
| MortgageDomain.xml | Declares what classes and attributes are available to PE. |
| PowerEditorConfiguration.xml | Configuration file for PE UI and server. |
| TemplateDefinition-Parameter.xml | Declares parameter templates. |
| county-enumeration-sample.xml | Defines U.S. county names for use with template columns. |
| State-enumeration-sample.xml | Defines U.S. state names for use with template columns. |

Figure 3‑1: PowerEditor Configuration Files

1. Customize the MortgageDomain.xml. You may want to use the MindBox DomainEditor[[1]](#footnote-1) to customize them or create them from scratch. You can rename this file.
2. If you plan to use parameters, update the TemplateDefinition-Parameter.xml file. See *PE Parameter Editor Developer Guide* for more details. You can rename this file.
3. If you plan to use U.S. state names or county names, update the state-enumeration-sample.xml and county-enumeration-sample.xml, respectively, as needed. If you need to define other enumeration values, create additional XML files that conform to the <install-dir>/xsd/XMLEnumeration.xsd schema.
4. Customize the PowerEditorConfiguration.xml file. For detailed discussion of each configuration parameter, see *Section* *6: PowerEditor Customization Guide.* 
   1. Modify the text of the <Deployment><BaseDir> tag so that it points to the directory where deployed rules will be stored. Make sure this directory exists.
   2. <Log> section defines fully qualified path names to PE log files. Make sure they point to a file in a directory that already exists.
   3. Modify <Server><KnowledgeBase><DomainFile> to point to the MortgageDomain.xml file you saved in Step 1 above. If you have more than one domain XML files, use additional <DomainFile> elements.
   4. If you have a parameter template definition file, modify reference to <Server><KnowlegeBase><TemplateFile> to point to the TemplateDefinition-Parameter.xml file you saved in Step 2.
   5. If you have an external enumeration XML files, specify them using <Server><EnumerationSources><EnumerationSource> elements. See Section 6.6 for details.
5. Set the PE config file path using ‘PEConfigFile’ Java System property. Typically, you add the following argument to a script that starts up application containers:

-DPEConfigFile=<absolute-path-to-config-file>

Different application servers provide different mechanisms for specifying System properties. Consult the application server’s documentation for details.

Alternatively, you can set an environment variable named ‘PEConfigFile’ to the absolute path to the PowerEditorConfiguration.xml file.

### Server Installation Step 4: Deploy PowerEditor WAR File

Deploy the PowerEditor WAR file into your application container. For IBM WebSphere, use the <install-dir>/webapps/powereditor\_websphre.war. For all other application containers, use <install-dir>/weapps/powereditor.war.

Consult your application container documentation for details on how to deploy WAR files. For example, for Tomcat, place the war file into the webapps directory.

Restart the Application Container and validate configuration as discussed in *Section 3.4* Server Configuration Validation and Troubleshooting.

## Server Installation – Database Configuration

This portion of the document describes database configuration steps for MS Access, MS SQLServer, MySQL, Oracle, and Sybase. Instructions in this section assume that the distribution zip file is extracted into <install-dir>.

As an overview, it is good to review that PowerEditorConfiguration.xml provides the following database options in the <Server><Database>. See the instructions for your particular database to see how to configure these options.

|  |  |
| --- | --- |
| Parameter | Value |
| Provider | Do not change. Should be com.mindbox.pe.server.db.DefaultPEDBCProvider |
| Driver | * For MS SQLServer, use “net.sourceforge.jtds.jdbc.Driver” * For MySQL, use “com.mysql.jdbc.Driver” * For ODBC (MS Access), use “sun.jdbc.odbc.JdbcOdbcDriver” * For Oracle, use “oracle.jdbc.driver.OracleDriver” * For Sybase, use “net.sourceforge.jtds.jdbc.Driver” |
| Connection | * For MS SQLServer, enter: “jdbc:jtds:sqlserver://<server>:<port>/<DatabaseName>” * For MySQL, use “jdbc:mysql://<server>:<port>/<DatabaseName>” * For ODBC Driver, enter: “jdbc:odbc:<DSN-Name>” * For Oracle Driver, enter: “jdbc:oracle:thin:@<server>:<port>:<DatabaseName>” * For Sybase, enter: “jdbc:pesybase:Tds:<server>:<port>/<DatabaseName>” |
| GuidelineRuleProviderClass | Only use for MS SQLServer and Oracle. For other types, leave it empty.  For MS SQLServer, use “com.mindbox.pe.server.db.SQLServerGuidelineRuleProvider”  For Oracle, use “com.mindbox.pe.server.db.Oracle9GuidelineRuleProvider” |
| User | Enter the login name for accessing the DB. |
| Password | Enter the *encrypted* password for the DB user. See Section *3.4 Password Encryption.* |
| MaxConnection | Enter the maximum number of concurrent connections allowed. Value greater or equal to ten is recommended. |
| ValidationQuery | This is an optional entry. Enter a valid select count query that should be run before using each DB connection to validate the connection, for example, select count(\*) from MB\_PRIVILEGE |

Figure 3‑ Database Configuration Options

### Microsoft Access DB Configuration

This section contains instructions for configuring a MS Access database with ODBC connector. The instructions assume that MS Access is already installed and running. The instructions also assume that the distribution zip file is extracted into <install-dir>.

##### Microsoft Access Database Configuration

The <install-dir>/database directory contains a blank MS Access database named MB-PEDB-EMPTY.mdb. You may use this for your installation. If you are using MS Access, simply rename this prior to use.

Note that the empty PE database contains only one user. Use demo/demo as the user id/password to sign in.

##### ODBC Configuration for Microsoft Access

In summary, the following steps create a new System ODBC DSN for the database setup in the above section.

1. Open Data Sources control panel, usually available under Control Panel / Administrative Tools.
2. Go to System DSN Tab, and click Add under the System DSN tab.

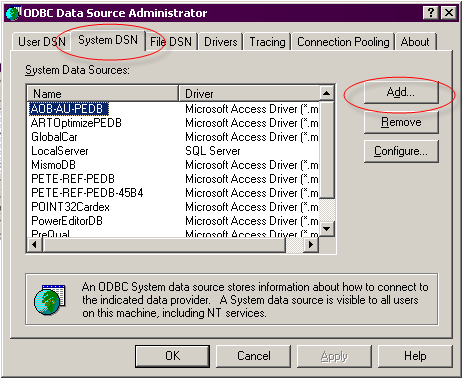


Figure 3‑ Create New ODBC Driver

1. Select the appropriate Driver type. Choose “Microsoft Access Driver (\*.mdb)”.

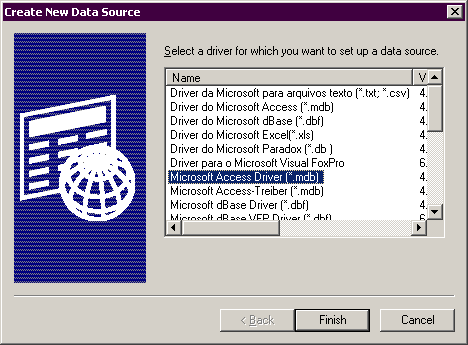


Figure 3‑ Create ODBC Driver for Microsoft Access

1. Follow the instructions on the screen to complete the ODBC setup.

Note that the default name to use for the DSN is “PowerEditorDB.” If you want to specify a data source name other than “PowerEditorDB,” please make a note of it, as you will need it when configuring the PowerEditor later.

### MySQL DB Configuration

This section contains instructions for configuring a MySQL database with JDBC connector. The instructions assume that MySQL is already installed and running. The instructions also assume that the distribution zip file is extracted into <install-dir>.

1. Determine the server name and port used by MySQL. By default MySQL runs on port 3306.
2. Create a new database for PowerEditor and a user who has create/select/insert/update access to the database.
3. Run all SQL script files in the <install-dir>/database directory whose name ends with ‘MySQL.sql’ or ‘generic.sql.’ Execute them in the order noted in the beginning of the filename. For example, execute Step1\_\* and then Step2\_\* etc.
4. Modify PowerEditorConfigruation.xml such that the <Server><Database> section reads as shown in the figure below. In this figure, in the <Connection> tag, host:port is the server name and port number used by MySQL, and database-name is the name of the database you created in Step 2. Be sure to update the <User> and <Password> tags with the correct value. See Section *3.3 Password Encryption.*

<Database>

<Provider>com.mindbox.pe.server.db.DefaultPEDBCProvider</Provider>

<MaxConnection>20</MaxConnection>

<!-- MySQL Settings -->

<Driver>com.mysql.jdbc.Driver</Driver>

<Connection>jdbc:mysql://host:port/database-name</Connection>

<User>powereditor</User>

<Password>9Ak9tSbRBG6GdVTvrpG7mQ==</Password>

</Database>

Figure 3‑: PE Configuration for MySQL

### Oracle DB Configuration

This section contains instructions for configuring an Oracle database with JDBC connector. The instructions assume that Oracle is already installed and running. The instructions also assume that the distribution zip file is extracted into <install-dir>.

1. Determine the server and port Oracle is running on.
2. Obtain the SID of the database to be used with PowerEditor.
3. Run all SQL script files in the <install-dir>/database directory whose name ends with ‘Oracle.sql’ or ‘generic.sql.’ Execute them in the order noted in the beginning of the filename. For example, execute Step1\_\* and then Step2\_\* etc.
4. Modify PowerEditorConfigruation.xml such that the <Server><Database> section reads as shown in the figure below. In this figure, in the <Connection> tag, host:port is the server name and port number used by Oracle, and SID is the SID of the PowerEditor database. Be sure to update the <User> and <Password> tags with the correct value. See Section *3.3 Password Encryption.*

<Database>

<Provider>com.mindbox.pe.server.db.DefaultPEDBCProvider</Provider>

<MaxConnection>20</MaxConnection>

<!-- Oracle Settings -->

<Driver>oracle.jdbc.driver.OracleDriver</Driver>

<Connection>jdbc:oracle:thin:@host:port:SID</Connection>

<User>mindbox</User>

<Password>9Ak9tSbRBG6GdVTvrpG7mQ==</Password>

</Database>

Figure 3‑: PE Configuration for Oracle

### MS SQLServer DB Configuration

This section contains instructions for configuring a MS SQLServer database with JDBC connector. The instructions assume that SQLServer is already installed and running. The instructions also assume that the distribution zip file is extracted into <install-dir>.

1. If using SQL server in a private development environment, configure SQL Server for *SQL Server and Windows* authentication. This will make it easier for PowerEditor to log into the SQL Server.
2. Change the SQL Server page size from 2k to 8k. This will allow more data per cell, which is needed for storing template definitions.
3. Determine the server and port SQL Server is running on. By default SQL Server runs on port 1433.
4. In SQL Server, create or identify a username that the PowerEditor will use to access the database.
5. Log into the SQL server with the same username that the PowerEditor will use for logging in. Logging in with the same username will make sure that table ownership is properly preserved.
6. Create a new database to be used with PowerEditor. Make sure that the user has the following permissions on all tables in the database: select, insert, delete, and execute.
7. Run all SQL script files in the <install-dir>/database directory whose name ends with ‘SQLServer.sql’ or ‘generic.sql.’ Execute them in the order noted in the beginning of the filename. For example, execute Step1\_\* and then Step2\_\* etc.
8. Modify PowerEditorConfigruation.xml such that the <Server><Database> section reads as shown in the figure below. In this figure, in the <Connection> tag, host:port is the server and port SQL Server is running on, and database-name is the name of the PowerEditor database created in Step 2. Be sure to update <User> and <Password> with the correct value. See Section *3.3 Password Encryption.*

<Database>

<Provider>com.mindbox.pe.server.db.DefaultPEDBCProvider</Provider>

<MaxConnection>20</MaxConnection>

<!-- SQL Server Settings -->

<Driver>com.microsoft.jdbc.sqlserver.SQLServerDriver</Driver>

<Connection>jdbc:microsoft:sqlserver://host:port;SelectMethod=Cursor;DatabaseName=database-name</Connection>

<User>servlet</User>

<Password>9Ak9tSbRBG6GdVTvrpG7mQ==</Password>

</Database>

Figure 3‑: PE Configuration for SQL Server

### Sybase DB Configuration

This section is used if you intend to use a Sybase database. The instructions assume that Sybase is already installed and running. The instructions also assume that the distribution zip file is extracted into <install-dir>.

Known Issue: PowerEditor 5.5.0 was the last version that was tested with Sybase. If you intend to use Sybase, please contact MindBox Technical Support.

1. Determine the server and port Sybase DB is running on.
2. Create a Sybase instance with 8KB page size. If you are using an existing instance, make sure the page size is greater than or equal to 8KB.
3. Create a new database to be used with PowerEditor with the minimum size of 10MB. If you are plan to import guidelines from a previous version of PowerEditor, increase the size accordingly.
4. Run all SQL script files in the <install-dir>/database directory whose name ends with ‘Sybase.sql’ or ‘generic.sql.’ Execute them in the order noted in the beginning of the filename. For example, execute Step1\_\* and then Step2\_\* etc.
5. Modify PowerEditorConfigruation.xml such that the <Server><Database> section reads as shown in the figure below. In this figure, in the <Connection> tag, host:port is the server and port Sybase DB is running on, and database-name is the name of the PowerEditor database created in Step 2. Be sure to update <User> and <Password> with the correct value. See Section *3.3 Password Encryption.*

<Database>

<Provider>com.mindbox.pe.server.db.DefaultPEDBCProvider</Provider>

<MaxConnection>20</MaxConnection>

<!-- SQL Server Settings -->

<Driver>com.mindbox.pe.wrapper.sybase.PESybaseDriver</Driver>

<Connection>jdbc:pesybase:Tds:host:port/database-name</Connection>

<User>servlet</User>

<Password>9Ak9tSbRBG6GdVTvrpG7mQ==</Password>

</Database>

Figure 3‑: PE Configuration for Sybase

### Table Indexing

If you are having performance issues, you'll probably want to profile your PowerEditor database. Here are some suggestions for DB indices based on previous profile runs:

MB\_ENTITY\_CATEGORY\_LINK: entity\_id and entity\_type (compound index)

MB\_ENTITY\_CATEGORY\_PARENT: category\_id and category\_type (compound index)

MB\_ENTITY\_GRID\_CONTEXT: grid\_id

MB\_GRID\_CELL\_VALUE: grid\_id

MB\_TEMPLATE\_COLUMN: template\_id

MB\_TEMPLATE\_COLUMN\_ATTR\_ITEM: template\_id

MB\_TEMPLATE\_COLUMN\_ENUM: template\_id

MB\_TEMPLATE\_DEPLOY\_RULE: template\_id

MB\_TEMPLATE\_DEPLOY\_RULE: template\_id and column\_no (primary key)

MB\_TEMPLATE\_MESSAGE: template\_id

MB\_TEMPLATE\_MESSAGE: template\_id and column\_no (compound index)

MB\_TEMPLATE\_MESSAGE\_FRAGMENT: template\_id

## Password Encryption

The PowerEditor database password and the LDAP server password, if LDAP is used, must be encrypted and entered into the PE configuration file before the PE server will start. A password encryption tool is included for completing this task. (This will prevent users with file system access from collecting these passwords.)

To retrieve and launch the tool:

1. Navigate to where PowerEditor tools are installed (typically, <install-dir>/tools).
2. Double-click on the powereditor-pwd-encrypt-tool.jar file or use the shell command *powereditor-tool.bat* for Windows and *powereditor-tool.sh* for UNIX.

To use the tool:

1. Enter the clear password into the Password field, make sure ‘Password in Server Config’ is selected for Encryption Type, and click the Encrypt button.

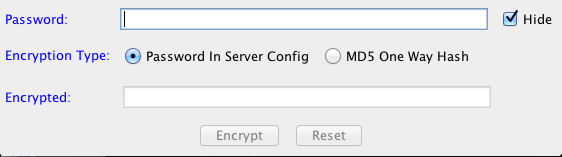
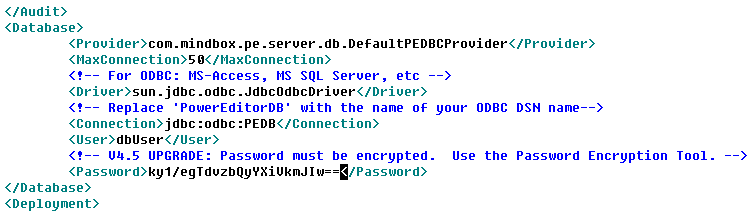


Figure 3‑ Password Encryption Tool

1. Copy the encrypted password into your PowerEditorConfiguration.xml file.



Click the Reset button to clear the fields if you make an error or have other passwords to encrypt.

If passwords aren’t encrypted the PE server instance will fail to start. (If this happens check the server logs or run http://<server>:<port>/powereditor/validate\_config.jsp to find out why.) If it’s an encryption issue you’ll see a message like this:

Failed to initialize PE Configuration from c:/mindbox/powereditor/config/PowerEditorConfiguration.xml:

Error decrypting password. Use the PowerEditorPasswordTool to encrypt passwords stored in config files.

## Server Configuration Validation and Troubleshooting

PowerEditor configuration can be complex. There is a tool to help validate and troubleshoot your configuration. The Validate Config tool is accessed by browsing to:

**http://***<server>***:***<port>***/powereditor/validate\_config.jsp**

This page validates server configuration. If it finds an error, it will be displayed on the screen. If successful, it will displays configuration details.

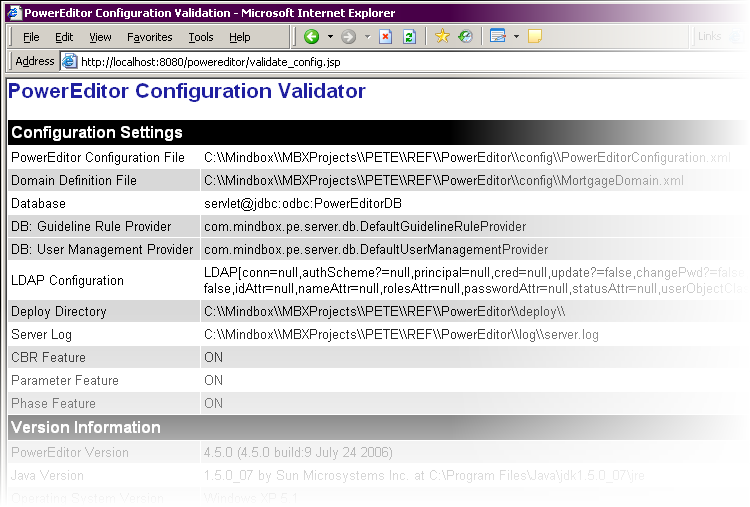


Figure 3‑ Successful Configuration Validation

Invalid configurations can manifest in different ways, often in generating an exception. Although an exception looks scary, the first line will often contain useful information. The following is an example of an invalid configuration.

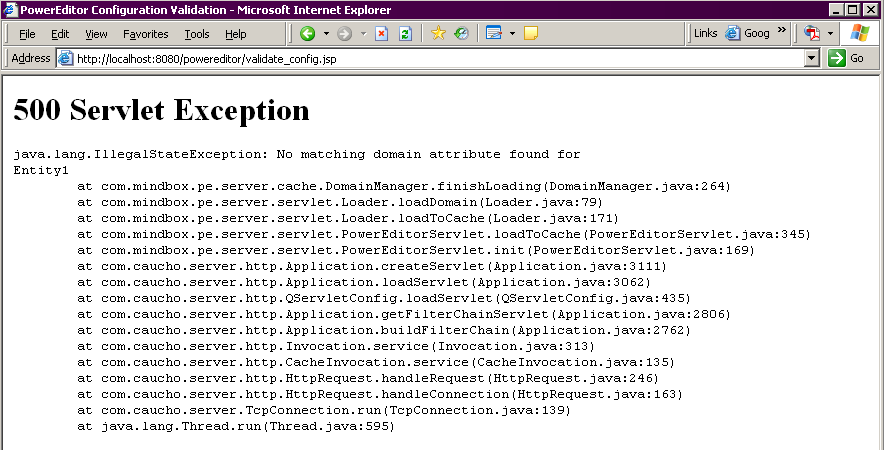


Figure 3‑ Example of Invalid Configuration

### Trouble-Shooting

If the config validation results in a browser page not found error, it means that the PowerEditor servlet could not be found. In this case, verify the following:

* Verify the application container started without any errors. You can do this by looking at the container’s log file. Each container vendor has separate locations for their proprietary log file.
* Verify that the container is listening to the port you specified. Again, each container vendor has separate locations specifying the port number.
* Verify that the host is accessible. This can be achieved by opening a DOS window, and typing:

**ping** *<host>*

* Look at the log file for the application container. Note that the PowerEditor log files will not be useful, since the PowerEditor was not reached.

If the browser shows a servlet exception, the PowerEditor was reached, but there was a configuration issue.

* Look at the error on the screen to see if that helps narrow down the configuration error.
* Look at the PowerEditor log files on the PowerEditor server. See *Section 7.5.5 Log Files* for a description of the log files
* Look at the Java console for client-side errors. See Figure 5-20 Checking Java Console for Client Configuration Error.

# PowerEditor Server Upgrade Procedures

Each version of PowerEditor comes with Release Notes that describes specific upgrade instructions. This section describes generic instructions for installing new versions of the PowerEditor over an old version.

1. Backup Old Configuration:
   1. Export all of your data from your existing configuration. Even for minor upgrades, this is a good process to follow. To do this, use the Admin: Export Data tab. Make sure all the checkboxes are checked (use the **Select All** button).
   2. Backup the PowerEditor database and all configuration files, including the domain definition (MortgageDomain.xml), the external enumeration XML flies, template definition (TemplateDefinition-Parameter.xml), and PowerEditorConfiguration.xml files.
2. Backup old PE Server Software version:
   1. Shut down the PowerEditor server.
   2. Backup the PowerEditor WAR file.
3. Install new server software version
   1. Obtain the PowerEditor distribution file and extract it into a temporary directory. We shall refer to this as <temp-dir>.
   2. Deploy <temp-dir>/webapps/powereditor.war to your application container.
   3. Start the application container, (If it was running, close and restart it). Your Servlet engine should expand the WAR file into a proper directory structure under the “<webapps>/powereditor” directory.
   4. Close down your servlet engine.
4. Update configuration files:
   1. If you have your own copy of LabelsBundle\_en.properties in your old powereditor directory under webapps, place a copy of that file in your new powereditor directory.
5. If you are upgrading to a newer version of the PowerEditor, read the release notes to ascertain whether any of your configuration files or database need to be updated to be compatible with the new version.

# PowerEditor Client Installation

## PE Client System Requirements

The PowerEditor is a Java application that can be accessed using *JNLP*. The following are requirements for running the PowerEditor.

* The client machine must have Sun Java JRE Version 1.6 or later. The Java JDK (a.k.a. the SDK) would also suffice, but is overkill. You can download the JRE from <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.
* The minimum recommended monitor display resolution is 1024x768 (A higher resolution is fine).
* The client machine should be configured with at least 1024 GB of RAM.

## PE Client Installation Instructions

### Basic Instructions

* Make sure that the client machine meets the system requirements above, including an installation of Java JRE.
* To display the PowerEditor login screen type the following into a command window:

**javaws http://***<host>***:***<port-number>***/powereditor/PowerEditor.jnlp**

Where:

*<host>* The machine on which the PowerEditor server resides. This could be a logical host name or an IP address. In the case that you are running the client on the same machine as the server, the host could be the literal localhost

*<port-number>* The port on which the Java servlet is configured. This is often 8080.

Examples:

javaws http://dev-machine:8080/powereditor/PowerEditor.jnlp

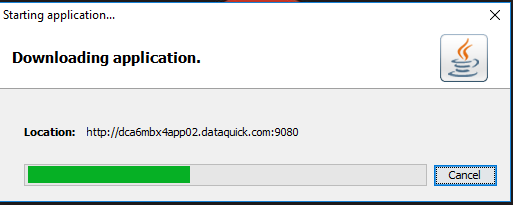
javaws http://dev-machine:3036/powereditor/PowerEditor.jnlp

javaws http://172.168.17.5:8080/powereditor/PowerEditor.jnlp

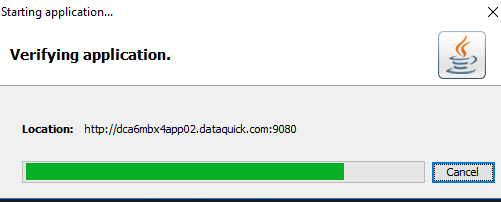
javaws http://localhost:8080/powereditor/PowerEditor.jnlp

You’ll probably chose to put this text within a batch file that executes this command. Place this batch file on a user’s desktop (any location will work) and have them use it to start the Power Editor.

A temporary dialog will display while the PowerEditor is downloaded (your URL will differ)



and another while it is validated.



It is possible the download and verification are fast enough so that you may not see either of these dialogs.

Figure 5‑ Application Download Dialogs

* At this point, you should see the PowerEditor login screen, shown below. The PowerEditor database is initially configured with one demo user and one admin user. You might want to login using admin user to create additional users and assign roles to them.
  + Username of demo user: demo
  + Password of demo user: demo
  + Username of admin user: admin
  + Password of admin user: admin



Figure 5‑ PowerEditor Login Screen

### Client Installation Refinements

Here are some suggestions to help fine-tune certain client installations, ensuring that the PowerEditor window is not occluded or stomped on.

* Sometimes PowerEditor client performance can be tuned by increasing the amount of RAM pre-allocated to Java applications as the application is launched. In particular, if your PowerEditor feels very sluggish while others have no troubles, it could be a RAM allocation issue. This can be configured in your Java installation. It is recommended that you increase this to at least 256 MB.
  + Go to your Windows Control Panel, and choose *Java*. The resulting dialogs are shown in the figures below. In the *Java* tab, under *Java Applet Runtime Settings,* choose the *View* button. Add the following line to the *Java Runtime Parameters:*

-Xmx256m

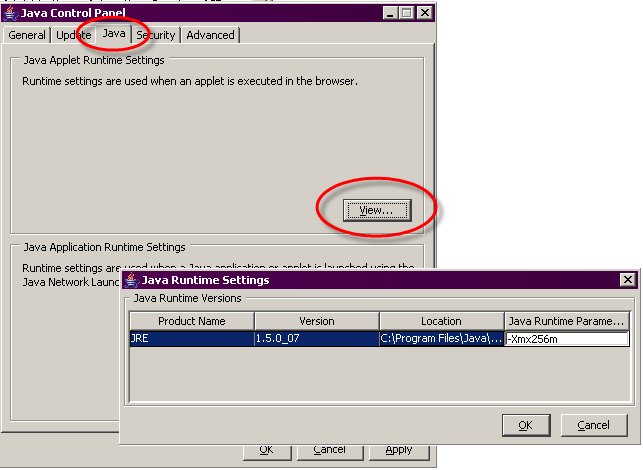


Figure 5‑ Java Control Panel for Client Performance

### PE Client Installation Trouble-Shooting Suggestions

This section highlights potential difficulties in installing a PowerEditor client. If you find that this section does not address your issues, please contact MindBox Technical Support.

#### Connection Failure

If the PowerEditor login screen (Figure 5‑13) does not appear, try some of the following:

* Try running the PE Validate Config tool from the client machine. Instructions for this are in *Section 3.4 Server Configuration*. In short, type the following in the browser:

**http://***<host>***:***<port-number>***/powereditor/validate\_config.jsp**

* Make sure you can get to the host by pinging the host name. From a DOS window, type

**ping** *<host>*

* If a new window is created, but has completely blank contents, it probably means that Java is not installed properly on the client machine. See Section 5.1 for more information. If you see the below screen after hitting the “Sign In” button, you can feel more confident that Java is installed.

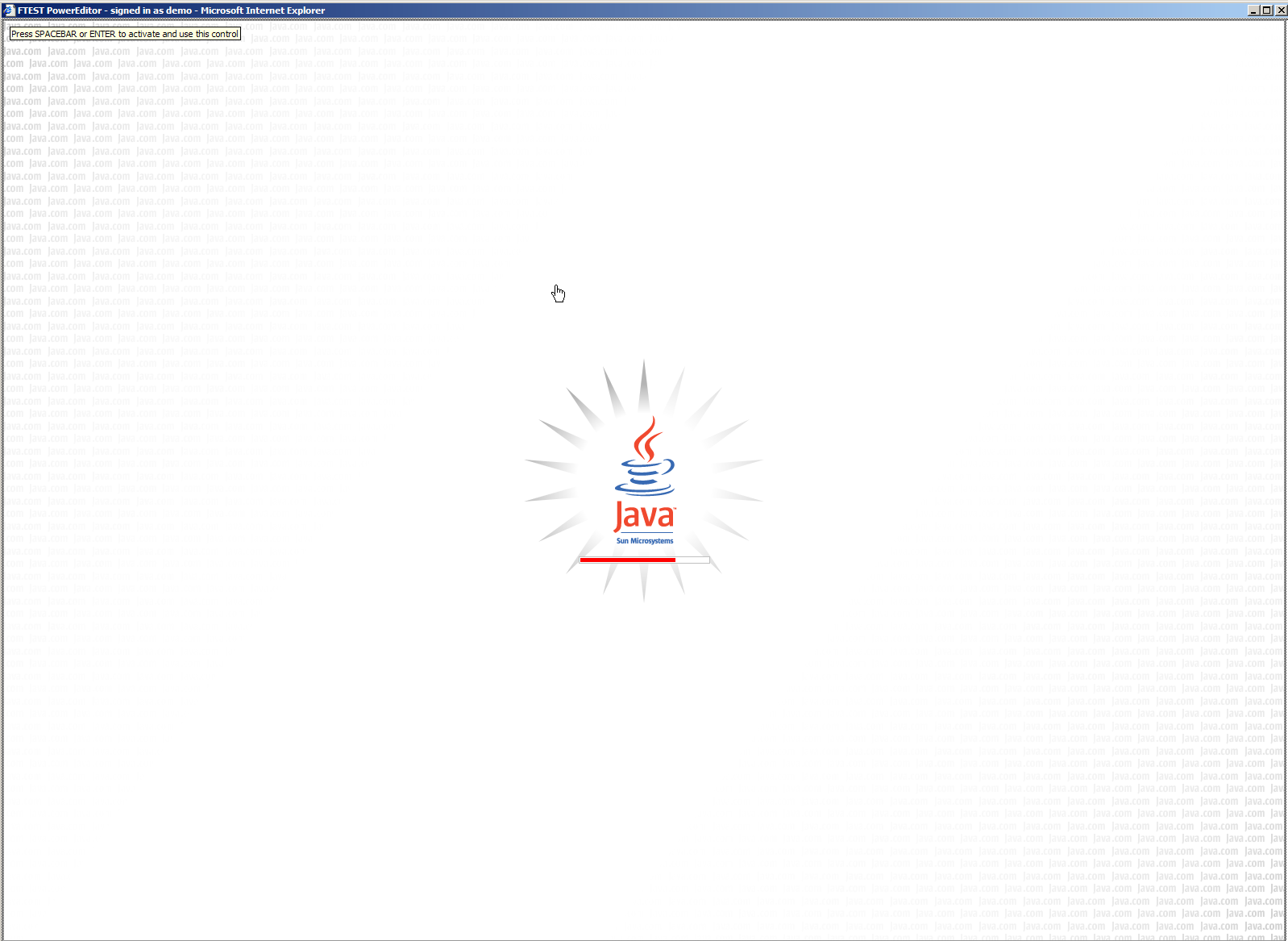


Figure 5‑ Indicates Proper Java Installation

* If you are really stuck, sometimes it is helpful to look at the log files on the server. *Section 3.5 Server Configuration* contains information about viewing log files.

#### Login Failure

Here are some issues that can occur after you’ve entered a username and password in the login screen (Figure 5-13).

* A login failure can sometimes be a symptom of server configuration issues: be sure to validate your configuration as described in *Section 3.5 Server Configuration*.



Figure 5‑ Potential Server Configuration Failure

* After logging in, the PowerEditor will create another window that will contain the applet. If that window is created but remains blank. In this case it might be useful to check the Java Console as is shown below. In the Java Console, look for errors:

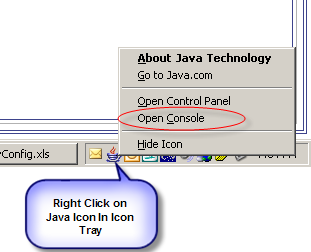


Figure 5‑ Checking Java Console for Client Configuration Error

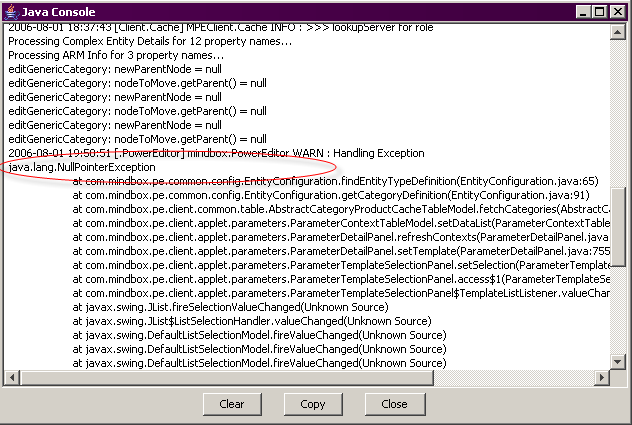


Figure 5‑ Java Console with Uncaught Configuration Error

#### Where are my Tabs?

If you find that the number of tabs are fewer than you expect it's probably because the user you logged in doesn't have expected permissions. Try logging in as admin/admin, then go to Admin->ManageUsers or Admin->ManageRoles.

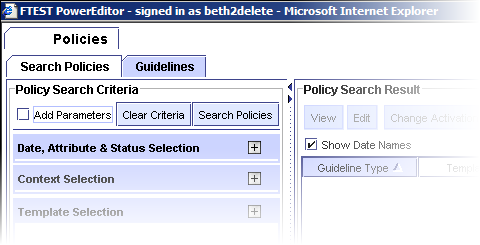


Figure 5‑ Login Result if User has no Roles or Privileges

# PowerEditor Customization Guide

The PowerEditor includes many options for customizing the tool to suit the needs of each installation of the tool. For example, entities can be renamed, tabs can be turned off, and rule generation can be configured for specific rule engines. The following two chapters present the many options that the PowerEditor provides for customization.

The first of these two chapters, *Chapter 6: PowerEditor Customization Guide* serves as a relatively in-depth discussion of each of the customizable options. It includes extensive verbiage, diagrams and examples. The subsequent chapter, *Chapter 7: Configuration File Reference* presents a table summary of configurable options.

The configurable properties of the PowerEditor are maintained in a centralized location: the *PowerEditorConfiguration.xml* file. This file is located in the <*install-dir>/config* directory. This file is sometimes loosely referred to as “***the config file***”. All configuration options presented in the next two chapters are from this file, unless otherwise noted.

## Path to PowerEditor Configuration File

PowerEditor configuration file can be specified via an environment variable or Java System property in addition to the [web.xml](http://web.xml) file. It is recommended that you use Java System property. This eases deployment process by eliminating steps to update [web.xml](http://web.xml). The name of the environment variable and the Java System property is the same: **PEConfigFile**.

To set the PE config file path using Java System property, add the following argument to a script that starts up application container:

-DPEConfigFile=<absolute-path-to-config-file>

Different application containers provide different mechanisms for specifying System properties. Consult the application container’s documentation for details.

## KB Filtering With Date Range

PowerEditor supports filtering the entire PowerEditor KB based on a date range. This allows multiple instances to use the same DB, given that no two instances use overlapping date ranges. In such scenario, only a single instance of PowerEditor is allowed to make changes to the KB. If an instance is configured with a KB filter date range with an end date, PowerEditor will run in read-only mode.

This section describes how to configure filtering date range source and provides a typical example of use.

### Configuration of KB Date Filter

To configure KB date filter, the following code should be added to the <PowerEditorConfiguration> element within the PowerEditorConfiguration.xml:

<KnowledgeBaseFilter>

<DateFilter>

<BeginDate>2009-10-01T00:00:00</BeginDate>

<!--

<EndDate>2010-12-31T23:59:59</EndDate>

-->

</DateFilter>

</KnowledgeBaseFilter>

**Notes:**

* Both the <BeingDate> and <EndDate> must be specified in the format of YYYY-MM-DD’T’HH:MM:SS (XML Schema date-time format)
* If the <EndDate> is specified, PowerEditor will run in read-only mode.

### Sample Set up

This section shows a sample set up of KB date filter for two PowerEditor instances that use the same database. The first instance will be configured to use the parts of KB up to April 1, 2010, making it a historical instance, operating in read-only mode. The second instance will be configured to work with the parts of KB since April 1, 2010, including all future additions and modifications.

**Instance One (Historical)**

<PowerEditorConfiguration>

<KnowledgeBaseFilter>

<DateFilter>

<EndDate>2010-03-31T23:59:59</EndDate>

</DateFilter>

</KnowledgeBaseFilter>

</PowerEditorConfiguration>

**Instance Two (Current)**

<PowerEditorConfiguration>

<KnowledgeBaseFilter>

<DateFilter>

<BeginDate>2010-04-01T00:00:00</BeginDate>

</DateFilter>

</KnowledgeBaseFilter>

</PowerEditorConfiguration>

## High-Level Tab Configuration

There are some high-level tabs, a.k.a. features, which can be disabled for a configuration. These tabs are:

* Case-Based Reasoning
* Parameters (appears under Policies tab)
* Manage Process

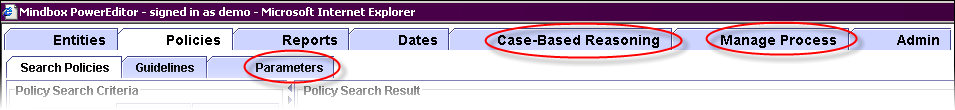


Figure 6‑ Configurable *Features*

To turn off a feature, set the enable attribute to “false” in the following section in the config file. Note that if a feature is not present in the config file, the feature will be enabled by default.

<FeatureConfig>

<Feature name="**cbr**" enable="false" className="com.mindbox.pe.client.applet.cbr.CBRPanel"/>

<Feature name="**parameter**" enable="true"

className="com.mindbox.pe.client.applet.parameters.ParameterManagerTab"/>

<Feature name="**phase**" enable="true"

className="com.mindbox.pe.client.applet.admin.ProcessManagementTab"/>

</FeatureConfig>

Figure 6‑ Feature Configuration Section of PowerEditorConfiguration.xml

Note that other tabs are also configurable, but appear in separate sections of the configuration file, and are therefore discussed later in this document. Such tabs include:

* Tabs that appear under the entity tab
  + Manage Products
  + Manage Channels
  + Manage Compatibility

Also note that some tabs may be disabled based on user permission. The mechanism for disabling tabs based on user permission is discussed in the document entitled *PowerEditor Reference Manual*. Examples of tabs that are disabled based on user permission are:

* Deploy
* Import/Export
* Manage Guideline Test Conditions
* Manage Guideline Actions
* Manage Roles
* Manage Users
* Reports

## Entity Configuration

### Entity Configuration Overview

*Entities* are objects that can appear in the context of guidelines. They represent basic central business concepts that are commonly associated with business policies. *Entity Types* are the *types* of objects that can appear in the context of guidelines. Examples of *entity types* in the mortgage domain include products, programs, channels and investors. For a more thorough conceptual description of entities, see the *PowerEditor User’s Guide.*

All Entity management functionality of the PowerEditor is available in the *Entities tab*. You may configure the Entities tab to contain any number of entity types. For each entity type, you can also configure the following:

* The *entity properties* that are available for each type of entity
* The *entity property tab* under which each property appears
* Whether each entity property is *searchable*
* Whether the entity property appears as a column in the *entity selection table*
* Whether *categories* are available for the entity type

The following figure shows how each of the configurable elements are displayed in the PowerEditor.

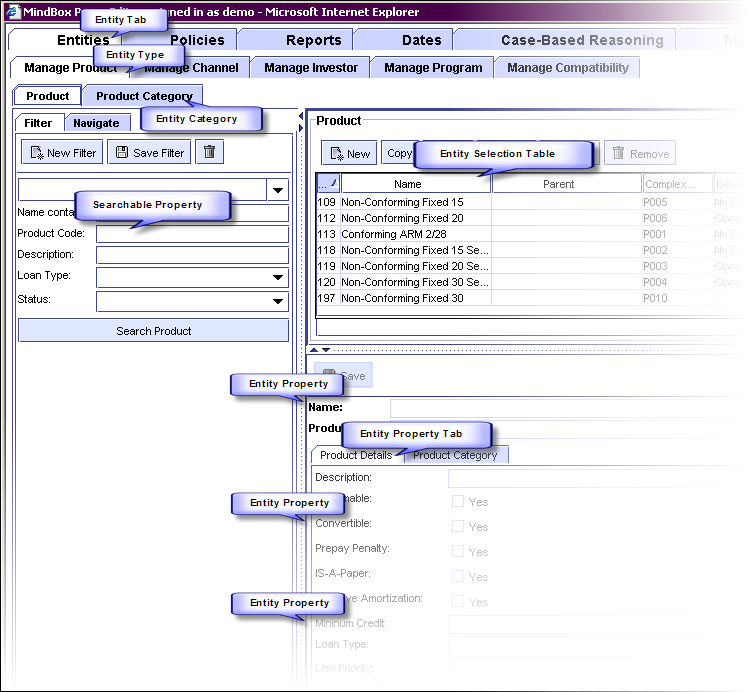


Figure 6‑ Configurable Entity Elements

The remainder of this section describes how to configure these entity elements. The PowerEditor separates entity type information into the areas listed below. In order to add a new entity, you need to create configuration elements in each area.

*1) Entity Definition:* Defines the category, entity, and properties, at the PowerEditor modeling level. This information is specified in the <EntityConfig> section of the config file. This section of the config file is documented in *Section 6.2.2 Entity Definition*.

*2) Entity Display* Defines which entity types and attributes will be displayed in the PowerEditor. This information is specified in the <UserInterface><Entity> section of the config file. This is documented in *Section 6.3.1 User Interface: Entity Display.*

*3) Entity Rule Mapping* Defines information relevant to how entities map into the Mortgage Domain objects at rule generation time. This information is specified in the <RuleGeneration><LHS><Pattern> section of the config file. It makes reference to information in the Domain file. This is documented in *Section 6.4.1.2 LHS Control Pattern.*

*4) Entity Parameter Mapping* Defines information relevant to how entities map into the Mortgage Domain objects for parameters at rule generation time. This information is specified in the <RuleGeneration>< ObjectGenerationDefault> section of the config file. It makes reference to information in the Domain file. This is documented in *Section 6.4.4 Parameter Generation Options*.

*5) Entity Domain Model* The definition of the entity and attributes in the Mortgage Domain file. This information is contained wholly within the Domain file.

Figure 6‑: Entity Definition Components

So, if you want to add a new entity to your configuration, and you’d like for that entity to be used in rule and parameter contexts, then you would need to modify each of the above components of your PowerEditor configuration file. It would probably be great to have a step-by step example of doing this, but that’s a great fantasy of documentation to come. For now, look at the section references listed in the box above.

### Entity Definition

As mentioned in the above overview, entity definition is achieved in the <EntityConfig> section of the config file. This section provides a conceptual overview of the EntityConfig section. For a detailed description of each element, see *Section 7.2: Summary of Entity Configuration Parameters*.

<EntityConfig>

<CategoryType name="**Product Category**" **typeID="10"** showInSelectionTable="Yes"/>

<CategoryType name="Program Category" typeID="40"/>

<EntityType name="**product**" displayName="Product" typeID="0" useInContext="Yes"

useInCompatibility="Yes" canClone="Yes" **categoryType="10"**>

<EntityProperty name="code" displayName="Product Code" showInSelectionTable="Yes"

isRequired="Yes" isSearchable="Yes" type="string"/>

<EntityProperty name="description" displayName="Description" showInSelectionTable="Yes"

isRequired="No" isSearchable="Yes" type="string"/>

<EntityProperty name="assumable" displayName="Assumable" showInSelectionTable="No"

isRequired="No" isSearchable="No" type="boolean"/>

</EntityType>

<EntityType name="program" displayName="Program" typeID="7" useInContext="Yes" useInCompatibility="Yes" canClone="yes" categoryType="40" >

<EntityProperty name="description" displayName="Description" showInSelectionTable="No"

isRequired="No" isSearchable="Yes" type="string"/>

<EntityProperty name="code" displayName="Code" showInSelectionTable="Yes"

isRequired="No" isSearchable="Yes"

type="enum" enumType="program.code" sort="Yes"/>

</EntityType>

<EntityConfig>

Figure 6‑ Sample EntityConfig Section of Config File

Three major elements are defined in the EntityConfig section: Categories, Entity Types, and Entity Properties.

##### CategoryType

This element defines a category, which will later be associated with an entity. To see an example of this association, see the bold items in Figure 6‑27 Sample EntityConfig Section of *Config File.*

If you want to have a category hierarchy associated with an entity type:

* Add a <CategoryType> element, with a typeID
* Add the typeID to the appropriate <Entity> categoryType element

To disable category support for an entity, do not supply a categoryType for that entity.

##### EntityType

This configuration element defines an entity type (e.g. Product, Program). It is used to specify how the entity type will be displayed and edited, including its display name, editable properties, and selection table. Specific configuration elements are as follows.

name The internal name of the entity type. This serves as an identifier for the entity type. This identifier is used throughout the config file when referencing the entity type.

displayName Name displayed in the PowerEditor UI.

typeID A unique ID.

useInContext Specifies whether this entity get displayed in the guideline context selection.

useInCompatibility Specifies whether this entity get displayed in the compatibility matrix

canClone Specifies whether the clone button is enabled for this entity. Cloning an entity puts the new entity in the context of all the guidelines that have the old entity

categoryType The typeID associated with the category hierarchy for this entity. This must match a value in CategoryType typeID

useInMessageContext Template messages can be associated with an entity type. For example, in mortgage applications, messages are sometimes associated with the *channel* entity type. For a given installation, messages can be associated with at most one entity type. This means that useInMessageContext can be set to yes for at most one entity type.

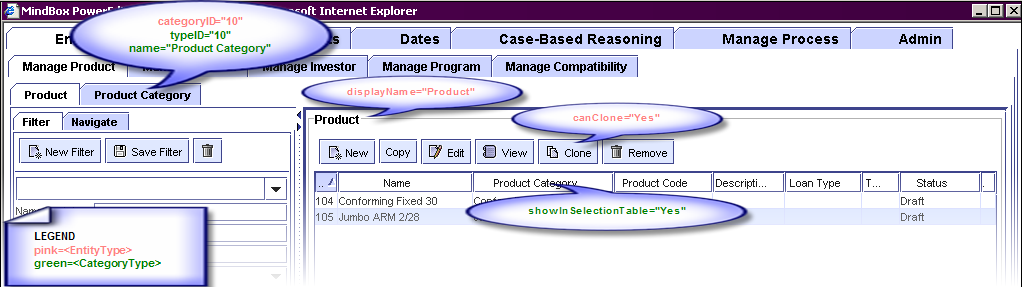


Figure 6‑ CategoryType and EntityType Elements

uniqueCategoryNames Specifies whether categories for this entity type should have unique names. If this is Yes, then the PowerEditor enforces unique category names for this entity type. See dialog below.

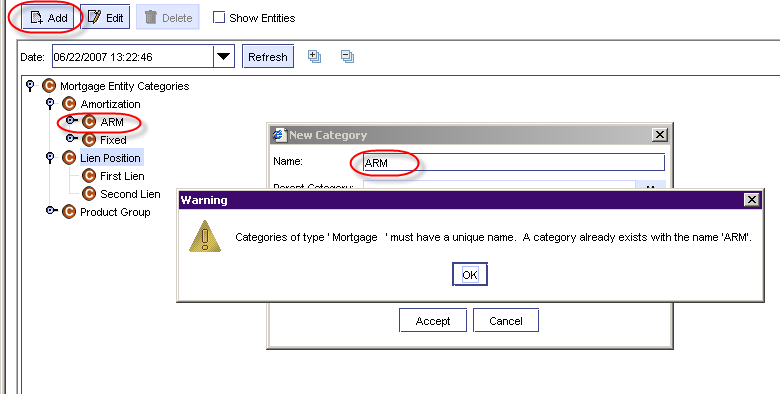


Figure 6‑ Creating Duplicate Node Names for Entity with *uniqueCategoryNames*

##### EntityProperty

The EntityProperty configuration element defines properties, or attributes, associated with a given EntityType. Entity Properties are editable fields that are displayed on the entity screen. See Figure 6‑23 Configurable *Features* for some example Entity Properties. You might notice that Entity Properties can appear within a Property Tab, or above the property tabs (as highlighted in Figure 6‑30 EntityProperty Elements). This is controlled in the <UserInterface> section of the config file, to be described in a later section.

Each PowerEditor EntityType comes predefined with two entity properties: *id* and *name.*

For each entity property, you can specify its field type, whether it is required, whether it appears in the selection table, and whether it appears in the search panel. These configurable elements are highlighted in Figure 6‑30 EntityProperty Elements*.*

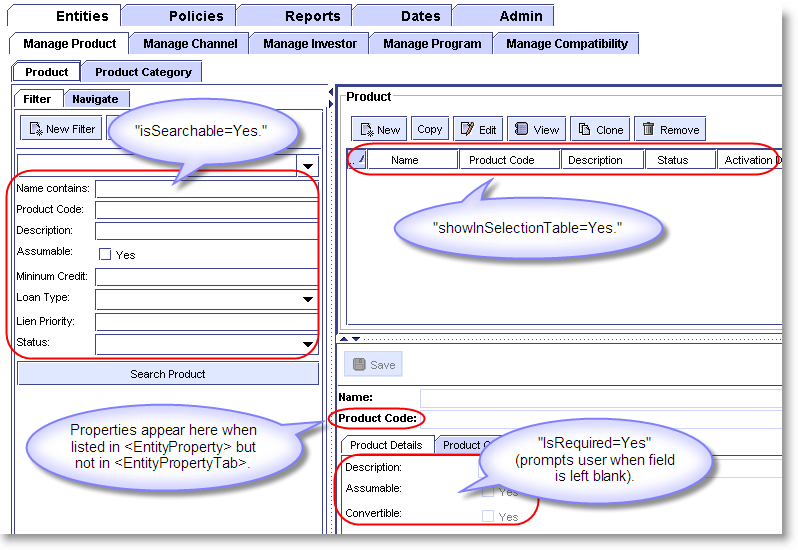


Figure 6‑ EntityProperty Elements

name The internal name of the property. This serves as an identifier for the entity type. This identifier is used throughout the config file when referencing the property. The value of this cannot be *id* or *name*, since these are predefined attributes automatically associated with each entity type.

displayName Name displayed in the PowerEditor UI.

showInSelectionTable Specifies whether this property is shown in the Product Selection pane.

isRequired Specifies whether the property's value must be filled in prior to saving. If the user attempts to save an entity without filling in a required field, they will receive an error. Required properties appear in bold.

isSearchable Specifies whether users can search by that property, and it appears in the Filter pane.

type The field type for the property. Valid values are: enum, string, boolean, symbol, integer, long, currency, percent, float, double, date.

enumType Valid only when type=enum and when attributeMap is not used. Specifies a database table in which enumerated values are listed.

attributeMap Valid only when type=enum and when enumType is not used. This specifies a pointer to an enumerated list that is in the domain file. This needs to be a *DomainClass.DomainAttribute*  reference. For example,

attributeMap=”LDO\_LOAN\_FEATURES.LoanDocumentationType”.

sort Valid only when type=enum. Specifies whether the PE should sort the enumerated values.

autoUpdatedDateProperty Valid only when type=date. Specifies that this date will be automatically updated whenever the specified property changes. For example, if autoUpdatedDateProperty=”Status” then whenever the Status property is edited, this date property will be automatically updated to the time that the Status property was changes.

allowMultiple Used only when type is “enum”. This allows multiple selection of values for the given property. This feature is added in version 5.6.0. Note that export of multi-select properties is written as a vertical bar (‘|’) separated list of values.

***Important Note***: As mentioned above, the value of the EntityProperty name cannot be *id* or *name*, since these are predefined attributes automatically associated with each entity type.



Figure 6‑ Illegal values for EntityProperty name

## User Interface Customization

This provides the ability to customize the PowerEditor User Interface for the specific application. The PowerEditor User Interface consists of two major components: Entity management and Guideline management. They are discussed below.

### User Interface: Entity Display

As mentioned in the Entity Configuration Overview(Section 6.2.1), there are 5 components used to define an entity. To review:

*1) Entity Definition:* Defines the category, entity, and properties, at the PowerEditor modeling level.

*2) Entity Display* Defines which entity types and attributes will be displayed in the PowerEditor.

*3) Entity Rule Mapping* Defines information relevant to how entities map into the Mortgage Domain objects at rule generation time.

*4) Entity Parameters Mapping* Defines information relevant to how entities map into the Mortgage Domain objects for parameters at rule generation time.

*5) Entity Domain Model* The definition of the entity and attributes in the Mortgage Domain file.

This section of the document describes the second component of entity configuration: Entity Display.

Entity Display is controlled in the <UserInterface><Entity> section of the PowerEditorConfiguration file. For a given entity type, you can specify which entity properties will be displayed, the order of the properties displayed, and the sub-tabs in which the properties will be displayed.

The following two figures show an example of configuring an entity display. For more details on the XML specification for the <UserInterface><Entity>, see *Section 7.3 Summary of UI Configuration Parameters.*

<UserInterface>

<Entity showTab="Yes">

<EntityTab type="product" showTab="Yes">

<EntityPropertyTab title="Product Details">

<EntityPropertyName>amortization.type</EntityPropertyName>

<EntityPropertyName>amortization.terms</EntityPropertyName>

<EntityPropertyName>interest.only</EntityPropertyName>

....

</EntityPropertyTab>

<EntityPropertyTab title="Arm Info">

<EntityPropertyName>arm.index.name</EntityPropertyName>

<EntityPropertyName>arm.first.adjust.period</EntityPropertyName>

<EntityPropertyName>arm.later.adjust.period</EntityPropertyName>

</EntityPropertyTab>

</EntityTab>

</Entity>

Figure 6‑ Sample Entity Section of Config File

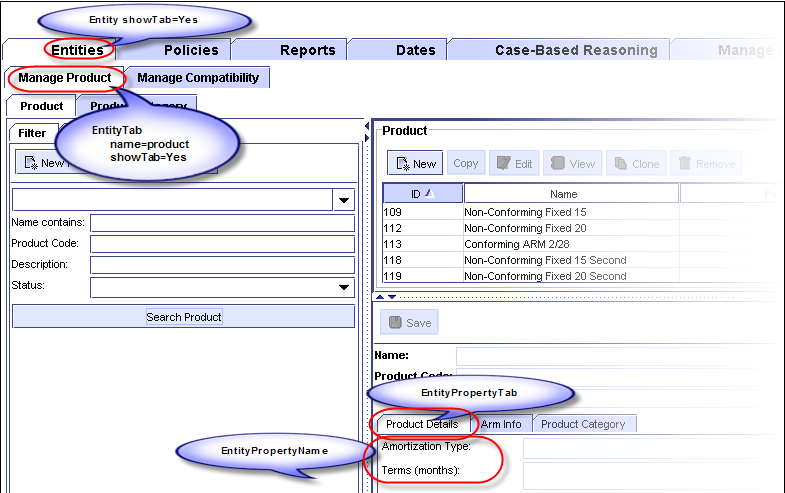


Figure 6‑ UserInterface Entity Elements

Note that properties specified in the EntityType that have showInSelectionTable="Yes" will show up in the Product details pane whether listed under an EntityPropertyTab or not.

### User Interface: Usage Type List

PowerEditor is configured with a set of usage types. Usage types are used to group like guidelines together, as well as, to provide context for phases. Each usage type is declared as <UsageType> tag in the <UserInterface><UsageTypeList> section. In addition, each usage type defined in the <UsageTypeList> must be included in the <UserInterface><GuidelineTab> declaration, in order to be recognized by the application. Refer to Section 6.3.3 for more details on the GuidelineTab section.

### User Interface: Guideline Management

#### Guideline Tabs

The <UserInterface><Guideline> section is used to define groupings of Usage Types within the Manage Guidelines screen.

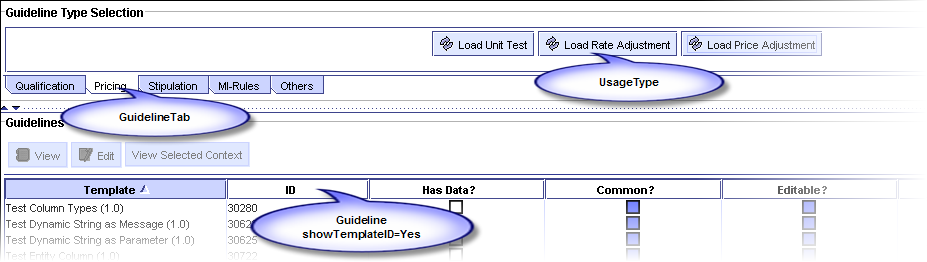


Figure 6‑ GuidelineTabs and UsageTypes

<UserInterface>

<UsageTypeList>

<UsageType name="Global-Qualify" displayName="Global Qualify" privilege="QualificationData"/>

<UsageType name="Product-Qualify" displayName="Product Qualify" privilege="QualificationData"/>

<UsageType name="Deal-Qual" displayName="Deal Qualification" privilege="QualificationData"/>

<UsageType name="Unit-Test" displayName="Unit Test" privilege="PricingData"/>

<UsageType name="Rate-Adjustment" displayName="Rate Adjustment" privilege="PricingData"/>

<UsageType name="Price-Adjustment" displayName="Price Adjustment" privilege="PricingData"/>

<UsageType name="Stipulation" displayName="Stipulation" privilege="QualificationData"/>

....

</UsageTypeList>

<Guideline showTemplateID="Yes">

<GuidelineTab displayName="Qualification">

<UsageType name="Global-Qualify"/>

<UsageType name="Product-Qualify"/>

<UsageType name="Deal-Qual"/>

</GuidelineTab>

<GuidelineTab displayName="**Pricing**">

<UsageType name="Unit-Test"/>

<UsageType name="**Rate-Adjustment**"/>

<UsageType name="Price-Adjustment"/>

</GuidelineTab>

....

</Guideline>

Figure 6‑ Sample UserInterface Guideline Section of Config File

#### Sorting Enumerated Values

Enumeration values of a single-select or multi-select enum column can be sorted or not, based either on the global configuration or the Sort Enum Values attribute associated with each column in the Guideline Template. To globally configure sorting, set the “sortEnumValue” attribute of the <Guideline> tag to “Yes”, as shown below:

<PowerEditorConfiguration>  
 <UserInterface>  
 <Guideline showTemplateID="Yes" **sortEnumValue**=”Yes”>  
. . .

To override the global setting on a column-by-column basis, use the column definition in the Guideline Template.

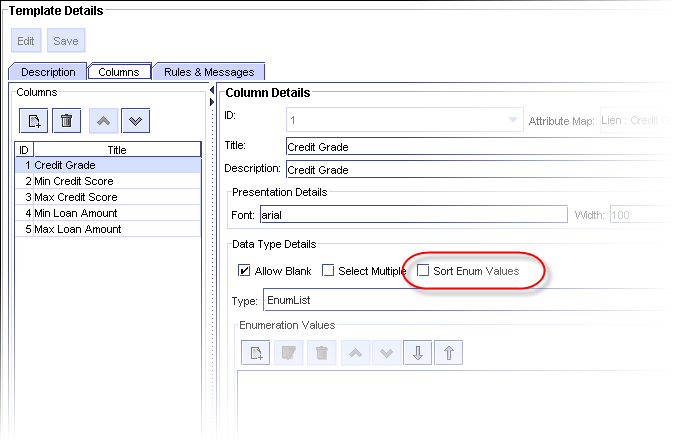


Figure 6‑ Overriding sortEnumValues

#### Fit Grids to Screen

PowerEditor, by default, displays grids in full size; the width of the grid is the sum of the width of each column in the grid. Typically, this sum is wider than the screen width. To accommodate this, you can instruct PE to shrink the grid to fit to the screen or not. To fit all grids to the screen width, set the “fitGridToScreen” attribute of the <Guideline> tag to “Yes”, as shown below:

<PowerEditorConfiguration>  
 <UserInterface>  
 <Guideline showTemplateID="Yes" **fitGridToScreen**=”Yes”>  
. . .

If this attribute is set to “No” or not specified, PE will display all grids in full width.

Note that each guideline template also has “fitToScreen” attribute. The value set in each guideline template overrides the option specified in the PowerEditorConfiguration.xml.

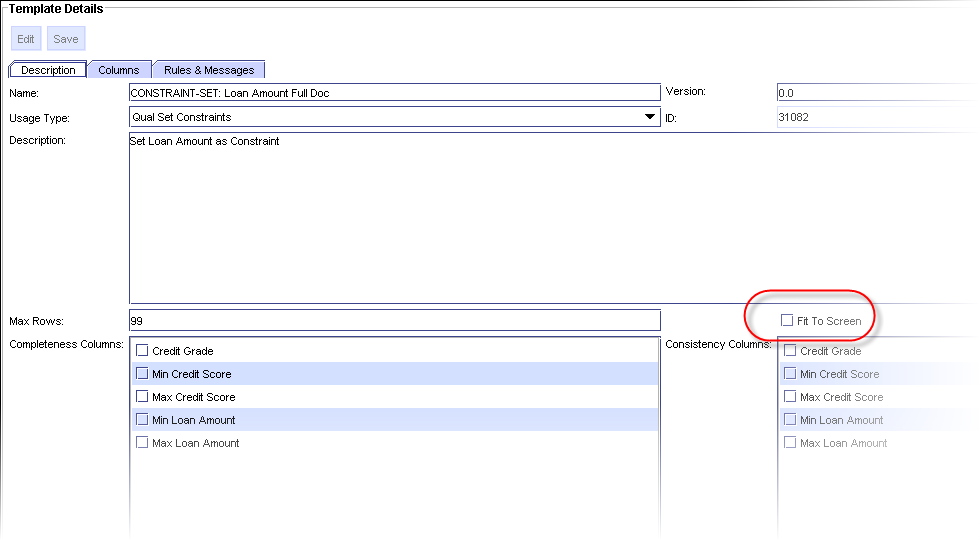


Figure 6‑ Overriding fitGridToScreen

### User Interface: Multiple Date Synonyms With Same Date

In the previous release, PowerEditor didn’t allow having more than one date synonym with the same date. That restriction can be removed in this release with a configuration change. To configure PowerEditor to permit more than one date synonym with the same date/time (TT-45), insert the following into the <DateSynonym> element of the <UserInterface> section:

<AllowIndenticalDates>true</AllowIndenticalDates>

### User Interface: Miscellaneous Options

The following provides a brief overview of other User Interface options:

<DateSynonym>

<DefaultTime>00:00</DefaultTime>

</DateSynonym>

<DeployExpirationDate>

<DefaultDays>30</DefaultDays>

</DeployExpirationDate>

<ClientWindowTitle>MindBox PowerEditor</ClientWindowTitle>

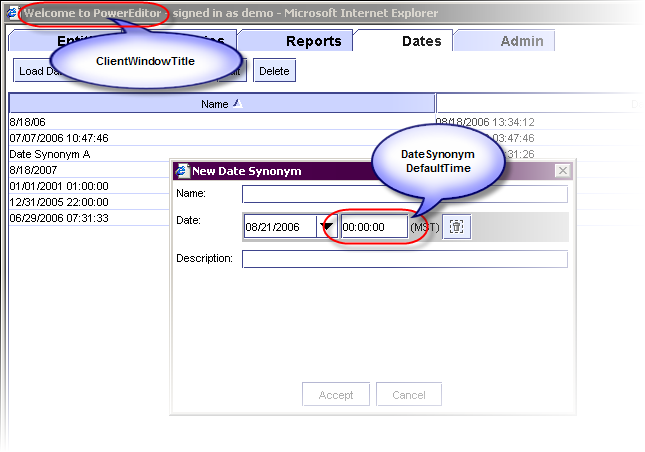


Figure 6‑ Date Synonym Default and Client Window Title Options

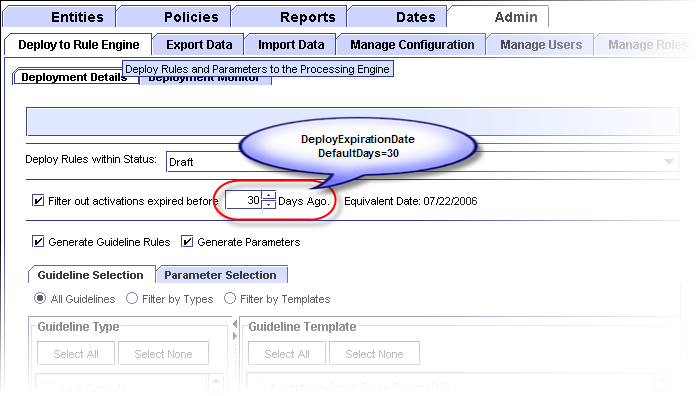


Figure 6‑ Deploy Expiration Date Option

### User Interface: Policy Management

#### Enforcing Sequential Activation Dates

PowerEditor gives you the option to enforce that all activation dates for a template and context are contiguous are non-overlapping.

 <UserInterface>  
  <UIPolicies>  
    <**EnforceSequentialActivationDates**>Yes</EnforceSequentialActivationDates>  
   </UIPolicies>

…

### User Interface: Show User’s Name in the Title Window

By default, PowerEditor displays the user’s ID in the title bar of the PowerEditor Window. To display user’s name instead, insert the following line inside the<UserInterface> element:

<UserDisplayNameAttribute>NAME</UserDisplayNameAttribute>

### User Inteface: Enable/disable User Account Functionality

PowerEditor has a feature for disabling and enabling user accounts. Only users with the admin privilege can enable or disable users. This feature is turned off by default. To turn this feature on, insert the following line in inside the <UserInterface> element:

<AllowDisableEnableUser>false</AllowDisableEnableUser>

### User Inteface: Access Disclaimer on Login Screen

Login screen displays ‘authorized users only’ access disclaimer message. The exact message displayed can be changed. Set the <UnauthorizedAccessWarningText> element with the desired message in the <UserInterface> section, as follows:

<PowerEditorConfiguration>

...

<UserInterface>

...

**<UnauthorizedAccessWarningText>The access to and use of the application is restricted to authorized users only.</ UnauthorizedAccessWarningText>**

### Failed Login Attempts Tracking

The counter that keeps track of invalid login attempts will be reset after a certain period of time. This allows users to reattempt authentication after the configured period of time has elapsed since last failure. This reset does not clear a lockout status.  Once a user is locked out they must be manually cleared.

Use the resetIntervalMins attribute of the <Lockout> element, which is in the <UserPasswordPolicies> element in the <Server> section, as follows:

<PowerEditorConfiguration>

...

<Server>

...

<UserPasswordPolicies>

**<Lockout resetIntervalMins=’60’/>**

## Rule Generation Options

The format of the ARTScript rules generated by the PowerEditor is controlled within the Rule Generation section of the configuration file. Rule generation options can be specified for the entire configuration using the <RuleGenerationDefault> element, and then can be overridden for each usageType. Separate elements are provided for LHS specification, RHS specification, message generation and parameter generation.

### Customize Number of Threads Used for Rule Generation

Deployment performance enhancement: different deployment artifacts are generated in different threads (TT-41). This reduces the overall deployment time. It’s possible to change the number of threads used for deploying guideline rules. By default, PE uses 8 threads. To change it, add the following with the desired value to the PowerEditorConfiguraiton.xml file, right below the <RuleGeneration> tag:

<GuidelineMaxThread>8</GuidelineMaxThread>

### Generate One Rule File Per Usage Type

PowerEditor now generates one guideline rule file per template by default. To have PE generate one rule file per usage type (to revert to the behavior of previous versions), add the following to the PowerEditorConfiguraiton.xml file, right below the <RuleGeneration> tag:

<MergeRuleFilesByUsageType>true</MergeRuleFilesByUsageType>

### Template Precision Field Use

Guideline template column has Precision field. The field has been used to determine decimal places shown in the policy grid. It can also be used to determine decimal places used when deploying currency/float/percent data, including range types. By default, the use of precision field to format values in the generated rules is disabled. This is to simulate behavior of previous versions.

To enable the use of precision field to format values in the generated rules, add the following to the PowerEditorConfiguraiton.xml file, right below the <RuleGeneration> tag:

<IgnorePrecision>false</IgnorePrecision>

### LHS Options

The LHS of each AE rule generated by the PowerEditor begins with a *request pattern* and a *control pattern*. The request pattern is the first pattern in the rule and describes the state of the engine in which the rule is intended to fire. The control pattern follows the request pattern and describes the contextual objects against which the rules fire. These patterns are each described below.

#### LHS Request Pattern

As mentioned above, the request pattern appears on the LHS of all rules generated by the PowerEditor. This pattern contains information about the state that the engine needs to be in for the rule to fire. For example, it contains information like the applicable focus of attention, and the date for which the rule applies.

The format of the request pattern is controlled by the following line in the LHS section of the configuration file:

<Pattern type="request" generate="Yes" class="phase" prefix="pa:" usageTypeAsFocus="Yes"/>

Figure 6‑: Configuration XML for Request Pattern

type Type=”request” specifies that the request pattern is being configured by the remaining attributes.

generate Specifies whether the request pattern should be generated.

class The request pattern will match against objects which are instances of the given class. See the examples below.

prefix The class name and all attribute names of the request pattern will be preceded by this prefix.

usageTypeAsFocus The gross-level format of the generated request pattern is determined by the value of the “usageTypeAsFocus” parameter. This is shown in the examples below.

MMS Compatibility Note

For Compatibility with MMS, “usageTypeAsFocus” should be set to “Yes”

When the value of usageTypeAsFocus is “Yes”, or is not specified, the following request pattern is generated.

(define-rule <name> <doc-string>

(object ?request

(instance-of <REQUEST-PATTERN-PREFIX><REQUEST-PATTERN-CLASS>)

(<REQUEST-PATTERN-PREFIX>focus-of-attention **<USAGE-TYPE>**)

(<REQUEST-PATTERN-PREFIX>current-time-slice ?time-slice <TIME-SLICES>))

)

<control-pattern>

<object-patterns-from-template-rule>

)

Figure 6‑: Request Pattern Generated when UsageTypeAsFocus=*Yes*

If the value of the “usageTypeAsFocus” parameter is “No”, then the following general format is generated for the request pattern. Notice that in this case, a ruleset declaration is generated, and the value of the *focus-of-attention* attribute is a guideline tab name rather than the usage-type.

(define-rule <name> <doc-string>

(declare (ruleset <USAGE-TYPE>))

(object ?request

(instance-of <REQUEST-PATTERN-PREFIX><REQUEST-PATTERN-CLASS>)

(<REQUEST-PATTERN-PREFIX>focus-of-attention **<GUIDELINE-TAB-NAME>**)

(<REQUEST-PATTERN-PREFIX>current-time-slice ?time-slice <TIME-SLICES>))

)

<control-pattern>

<object-patterns-from-template-rule>

)

Figure 6‑: Request Pattern Generated when UsageTypeAsFocus*=No* (pre-MMS)

The following describe components of the generated rules above.

<USAGE-TYPE> The usage type of the template for which a rule is generated. The text generated here will be a symbol value in all caps. See *Section 6.3.1* for more on usage types.

<REQUEST-PATTERN-PREFIX> The prefix for the request pattern, as specified in the configuration XML for the request pattern. The prefix corresponds to the module in which the request pattern class resides. An example prefix is “pa:” or “pe:” Note that if a colon is desired between the prefix and the object/attribute name, then the colon needs to be specified as part of the request-pattern-prefix.

<REQUEST-PATTERN-CLASS> The class for the request pattern, as specified in the configuration XML for the request pattern.

<TIME-SLICES> The times during which the rule is active. These are based on the activation date and expiration date of the guideline as specified the PowerEditor GUI. See MMS PE 5 documentation for a description of time-slices.

#### LHS Control Pattern

Like the request pattern, the *control* pattern also appears on the LHS of all rules generated by the PowerEditor (unless control pattern generation is disabled). The control pattern describes the contextual objects against which the rules fire. Specifically, it contains the entities against which the rule applies.

(define-rule <rule-name> <doc-string>

<request-pattern>

(object ?<CONTROL-PATTERN-VARIABLE>

(instance-of <CONTROL-PATTERN-CLASS-DEPLOY-NAME>)  
 (<CONTROL-PATTERN-ENTITY-TYPE-DEPLOY-NAME> <ENTITY-MATCH>)  
 (<CONTROL-PATTERN-ENTITY-TYPE-DEPLOY-NAME> <ENTITY-MATCH>)

....

)

<object-patterns-from-template-rule>

)

Figure 6‑: Generated Control Pattern Found in Each Deployed Rule

The content of the generated control pattern is dependent on both the Configuration XML and the Domain XML file. Specifically, the Configuration XML provides pointers to the object and attributes used in the control pattern, while the Domain XML defines the actual object and attributes.

The following code fragments show a sample generated control pattern, followed by a set of sample configuration files that generate this pattern. In this example, the control object is referenced as follows:

* mms:deal is the AE name of the control object class. The control object is an instance of this class.
* The name of the variable ?peDeal references the PowerEditor name of the control object class.

Similarly, for control attributes:

* mms:product, mms:channel, mms:investor,mms:program are the AE names of the control attributes.
* peProduct, peChannel, peInvestor, peProgram are the PowerEditor names of the control attributes.

(define-rule Rule1 “Template 1”

<request-pattern>

(object ?peDeal

(instance-of mms:deal)

(**mms:product** ?product &:(pe:entity-match :type Product :id ?product

:time-slice ?time-slice :category (build$ (3041))))

(**mms:program** ?program &:(pe:entity-match :type Fourth :id ?program

:time-slice ?time-slice :category (build$ (3344) (3345)))) )

<object-patterns-from-template-rule>

)

Figure 6‑: Sample Generated Control Pattern

The contents of the above control pattern are based on the configuration files which follow.

<RuleGenerationDefault>

<LHS>

<Pattern type="control" generate="Yes" class="peDeal">

<Attribute type="product" name="**peProduct**"/>

<Attribute type="program" name="peProgram"/>

</Pattern>

Figure 6‑: Sample Configuration XML for Control Pattern

In the above excerpt from the Configuration XML file

* The values of the “type” attribute must match the names of entities as specified in the EntityType section of the Configuration file.
* The values of the “name” attribute must be an attribute names that appear in the Domain XML file.

<DomainClass DeployLabel="mms:deal" DisplayLabel="deal" Name="peDeal" >

<DomainAttribute Name="**peProduct**" DeployLabel="**mms:product**" DeployType="Symbol"/>

<DomainAttribute Name="peProgram" DeployLabel="mms:program" DeployType="Symbol"/>

</DomainClass>

Figure 6‑: Sample Domain XML for Control Object

The above excerpt from the domain file is simplified for this example (e.g. the ClassKey and DisplayName attributes are omitted). This excerpt shows the actual definition of the object that acts as the control object. The important thing to point out is that, from the point of view of the domain file, the control object is no different than any other object in the Domain XML. This object is a control object only by virtue of the fact that it is referenced in the Configuration file’s control section.

**Control Pattern Completeness**

***Configuration File:*** Make the following changes to your pre-existing configuration file:

Every control pattern specification in the config file must now contain a reference to every entity type that is specified in your configuration file. If you don't want an attribute generated for a specific usage-type, the entity type must appear in the new *disallowedEntities* section of the control pattern.

In the below examples, there are 3 entity types defined for the configuration: product, channel and investor. In the first example, all 3 will be generated as part of the control pattern.

<Pattern type="control" generate="Yes" class="deal">

<Attribute type="product" name="product"/>

<Attribute type="channel" name="channel"/>

<Attribute type="investor" name="investor"/>

</Pattern>

In the next example, all only one entity type will be generated as part of the control pattern. Note that the other two MUST appear in the disallowedEntities element.

<Pattern type="control" generate="Yes" class="lien" **disallowedEntities="channel,investor"**>

<Attribute type="product" name="product"/>

</Pattern>

The PE generates a startup error if an entity type is missing.

#### LHS Static Pattern

You can insert a static pattern to all object patterns for instances of a class of which the name begins with a given string. This is called the “lineage” pattern. Use the <Pattern> tag with type “lineage”. For example, the following will have PE insert “(scenario-id ?scenario-id)” pattern into all object patterns whose class name begins with “ldo” or “bfe”.

<Pattern type="lineage" generate="Yes" prefix="ldo:,bfe:" text="(scenario-id ?scenario-id)" variable="scenario-id"/>

#### LHS Empty Pattern Variable Option

Empty patterns are attribute patterns that match on any value. You may configure the PowerEditor to prefix variable names for empty patterns with a dollar sign (‘$’). If variable names are prefixed with a dollar sign, the pattern will match even if the attribute has no value. However, such variables cannot be used in an arithmetic expression. So, if you are using arithmetic expressions, remember to turn this option off.

This option is specified in the “asSequence” attribute of the empty pattern tag, as follows:

<Pattern type="empty" generate="Yes" asSequence="true"/>

#### LHS - Setting Deploy Value For IS-Empty Operator

A guideline rule can have a condition checking whether an attribute is empty or not. Use “IS EMPTY” operator in PE template management screen. For these IS-EMPTY conditions, PE uses a predefined deploy value when generating object patterns for them. This deploy value is configured in the <LHS> of <RuleGeneration> section as follow:

<Value type="unspecified" deployValue=":UNSPECIFIED" valueAsString="No" />

In the above case, PE will generate an object pattern that will match on the symbol ‘:UNSPECIFIED’ for each IS-EMPTY condition.

### RHS Options

This section describes options for the Right Hand Side (RHS) of rules.

#### RHS Multi-Enum As Sequence Option

When the RHS of a guideline rule references a column that accepts multiple enumeration values, PE can either generate the value as an ARTScript sequence or a list of values, separated by vertical bars. The default is to generate them as sequences. To modify this option, use the following section:

<MultiEnumAsSequence>Yes</MultiEnumAsSequence>

Please note that if you set this option to No, you must make sure that the deploy type of the guideline action parameter that references the column is STRING. If not, PE will generate ARTScript that will not load in the engine.

### Message Generation Options

Message generation options are specified in the <MessageTypes> section of <RuleGeneration> section. For example,

<MessageTypes>

<Message type="range" rangeStyle="verbose"/>

<Message type="enum"

cellSelection="enumIncludeMultiple"

enumDelimiter=", "

enumFinalDelimiter=" or "

enumPrefix=" any of "/>

<Message type="enum"

cellSelection= "enumExcludeSingle"

enumPrefix=" not "/>

<Message type="enum"

cellSelection="enumExcludeMultiple"

enumDelimiter=", "

enumFinalDelimiter=" or "

enumPrefix=" not any of "/>

<Message type="conditional"

conditionalDelimiter=", "

conditionalFinalDelimiter=" and "/>

</MessageTypes>

Refer to PowerEditor-Messaging-DynamicStrings.doc and Template\_Message\_XML\_Specification.doc documents for more information on these options.

### Parameter Object Generation Options

PE provides a set of options to control how object instances from parameters are generated.

NOTE: As of PowerEditor 5.9.0, the <ObjectGenerationDefault> element must be placed as a child of <RuleGeneration> tag. Previous versions also accepted <ObjectGenerationDefault> tag inside the <RuleGenerationDefeault> tag, but current version will not accept it.

<ObjectGenerationDefault>

<InstanceCreateText>make-instance</InstanceCreateText>

<!-- For parameter context generation :

type refers to a context element entity type

name must match with a name attribute of a domain attribute

-->

<ParameterContext>

<Attribute type="product" name="product" value="code" valueAsString="Yes"/>

<Attribute type="investor" name="investor" value="id"/>

<Attribute type="channel" name="channel" value="id"/>  
 <Attribute type="activationDate" name="activationDate" value="id"/>

<Attribute type="expirationDate" name="expirationDate" value="id"/>

</ParameterContext>

</ObjectGenerationDefault>

In the above example, InstanceCreateText option specified the name of the function to use to create object instances. Valid values are make-instance and define-instance.

ParameterContext section defines attributes to use for setting context elements in the generated object instances. Note that the name attribute of each <Attribute> section must refer to an existing attribute of the domain class that parameter templates are mapped to.

## Server Customization

Server customization provides the ability to configure the operational aspects of the PowerEditor, such as logging, database connectivity, deployment, session control, etc. These appear in the *<Server>* section of the config file. See *Section 8.5 Summary of Server Parameters* for a summary of server options.

### Knowledge Base

The KnowledgeBase section of the server configuration specifies the location of knowledge base configuration files.

* DomainFile specifies domain attributes and classes. Multiple domain files can be specified for a single configuration.
* TemplateFile specifies parameter “templates”. Parameter templates specify the structure of parameter objects. Multiple template files can be specified for a single configuration.

<Server>

...

<KnowledgeBase>

<DomainFile>C:\\Mindbox\\powereditor\\MortgageDomainMISMO.xml</DomainFile>

<DomainFile>C:\\Mindbox\\powereditor\\MortgageDomainAO.xml</DomainFile>

<TemplateFile>C:\\Mindbox\\powereditor\\Pricing-Parameters.xml</TemplateFile>

<TemplateFile>C:\\Mindbox\\powereditor\\Qual-Parameters.xml</TemplateFile>

</KnowledgeBase>

### Audit Trail

PowerEditor creates audit logs in the MB\_AUDIT table. The audit logging can be turned on or off with the options of “Yes” or “No” as shown below.

<Server>

...

<Audit>

<AuditAll>**Yes**</AuditAll>

</Audit>

### Database Connection

The database connection information, including JDBC driver and connection string can also be configured. These options are described in the Installation section of this document.

### Deployment

Options for managing the rule deployment process are configured in the Deployment section of the PowerEditor configuration file. Note that the *structure* of deployed rules is not configured here, only the process. See *Section 6.4 Rule Generation Options* to configure rule structure.

<Server>

<Deployment>

<BaseDir>c:\\temp\\powereditor\\deploy\\</BaseDir>

<SaveOldFiles>Yes</SaveOldFiles>

<UseTimeStampFolder>Yes</UseTimeStampFolder>

<PostDeployScript>

<File>c:\\powereditor\\deploy\\sample-script.bat</File>

</PostDeployScript>

<ReportMissingLink>Yes</ReportMissingLink>

</Deployment>

* BaseDir specifies the directory in which the generated rule files are written.
* SaveOldFile indicates whether or not to save the previously generated rule files. If SaveOldFile=yes, a backup folder is created, and a time-stamped subfolder is created under that.
* UseTimeStampFolder indicates whether to place generated files to a time-stamp directory. This means that a unique directory would be created that included the creation timestamp in the directory name.
* PostDeployScript specifies the name of a file that contains a script that will be executed after the rules are deployed. This script will be run even if there are deploy errors. The script should be one that runs without any user intervention. For Windows, the script file is typically something like a .cmd or a .bat file.
* ReportMissingLink is used to turn off message generation for every missing link between parent-child classes that are in Exist conditions. This should generally be set to Yes, unless you are upgrading from PowerEditor 3.3 installations, in which case the messages can be extraneous.

#### Deployment Directories

The location of deployed files is dependent on the values of the following 3 configuration options, as well as the status selected at deploy time. The 3 configuration options that affect deployment location are: BaseDir, SaveOldFile, and UseTimeStampFolder.

For the sake of discussion, say that BaseDir="c:\PowerEditor\deploy". When rules are deployed, a folder is created underneath the BaseDir named after the status for which you are deploying. For example, if you deployed rules of status Draft, a directory is created called c:\PowerEditor\deploy\Draft.

If UseTimeStampFolder=no then all your generated files will appear underneath the status directory, e.g. c:\PowerEditor\deploy\Draft\pricing-rules.art. If SaveOldFile=no, any preexisting files will simply be overwritten. If SaveOldFile=yes, a new folder timestamped folder is created with the back, e.g. c:\PowerEditor\deploy\backup\Draft\2006-08-03-05.13.49\pricing-rules.art

If UseTimeStampFolder=yes, a timestamp folder will be created underneath the status folder, e.g. c:\PowerEditor\deploy\Draft\2006-08-03-05.13.49\pricing-rules.art. In this case, the SaveOldFile is basically ignored, since the timestamp scheme files from being deleted or overwritten.

That's all pretty straightforward. The trickier part is *which* rules get deployed with what status. PowerEditor thinks of status as a linear sequence, based on configured IDs. So for example, Draft is lower than QA is lower than Production. When deploying rules of a certain status:

* All rules of that status and above get deployed
* They all get deployed into the folder of the selected status
* All other folders remain unchanged.

Let’s take an example. If you deploy QA rules:

* The deployed rules will be comprised of QA and Production rules
* All of these rules will be placed in the QA folder
* The contents of the Production and Draft folders will remain unchanged.

#### Data Available to Post Deployment Script

This feature allows a script to be executed immediately following the generation of all the ART-Script rule files. You can specify any file that can be executed by the Operating System (e.g. .exe, .bat, .sh, .pl). The full pathname of the script file must be designated in the <PostDeployScript> section. For example:

<PostDeployScript>

<File>c:\\powereditor\\sample-script.bat</File>

</PostDeployScript>

PowerEditor passes deploy status and directory to post deploy scripts. PowerEditor passes them both as command line arguments and as environment variables. In addition, PowerEditor provides the ID of the user who initiated deployment process as third argument. It also sets an additional environment variable named ‘USER’ with the user’s ID.

The following table summaries data passed to post deploy scripts. Sample scripts are provided in the 'config' directory in the PowerEditor distribution file.

|  |  |  |
| --- | --- | --- |
| **Data** | **Command Line**  **Argument Position** | **Environment Variable**  **Name** |
| The deploy status the user specified for deployment. | 1 | KB\_STATUS |
| The full path of the deployment directory. This will include time stamp if use of time stamp is turned on. | 2 | DEPLOY\_DIR |
| The id of the user who initiated the deployment process. | 3 | USER |

### Log Files

The PowerEditor provides four log files. These are helpful for historical introspection, and also debugging. The four PowerEditor log files are as follows:

1. Loader - The loader log reports entries in the database that the PowerEditor server has successfully loaded into memory.
2. Deployer - The deployer log contains information related to deployment, including errors that occurred while deploying rules.
3. Database - The database log contains information related to database operation.
4. Server – The server log tracks events that happen in the PowerEditor servlet, other than those that occur during load, deploy, or database operations.
5. Servlet – The servlet log also tracks events that happen in the PowerEditor servlet, but at the request/response level.

The names, locations, of each of these files are configurable.

<Server>

...

<Log>

<LogFile type="**database**" writeLog="Yes" debug="No" maxSize="2"

file="c:\\temp\\powereditor\\log\\database.log"

pattern="%d{yyyy-MM-dd HH:mm:ss} [%7.15c] %c{2} %-5p: %m%n"/>

<LogFile type="**deployer**" writeLog="Yes" debug="Yes" maxSize="4"

file="c:\\temp\\powereditor\\log\\deployer.log"

pattern="%d{yyyy-MM-dd HH:mm:ss} %-5p [%c{2}] %m%n"/>

<LogFile type="**loader**" writeLog="Yes" debug="Yes" maxSize="4"

file="c:\\temp\\powereditor\\log\\loader.log"

pattern="%d{yyyy-MM-dd HH:mm:ss} %-5p [%c{2}] %m%n"/>

<LogFile type="**server**" writeLog="Yes" debug="Yes" maxSize="4"

file="c:\\temp\\powereditor\\log\\server.log"

pattern="%d{yyyy-MM-dd HH:mm:ss} %-5p [%c{2}] %m%n"/>

<LogFile type="**servlet**" writeLog="Yes" debug="Yes"/>

</Log>

PowerEditor supports 3 basic levels of logging:

* None – no logging
* Informational – moderate logging
* Debug - most verbose

These are set via the writeLog and debug attributes in the configuration file.

|  |  |  |
| --- | --- | --- |
| **To achieve this logging level:** | **Set debug to:** | **and set writeLog to:** |
| None | "No" | "No" |
| Info | "No" | "Yes" |
| Debug | "Yes" | "Yes" |

### Session Configuration

PowerEditor supports the following session configuration options: maximum concurrent sessions, session time-out limit, and SSO Support.

#### Maximum Concurrent Sessions

PowerEditor allows a limited number of concurrent sessions. The default value for this limit is 4. To change the limit, change the <Server><Session><MaxUserSessions> tag in the PowerEditorConfiguration.xml file.

#### Session Time-out

The default session time out is 30 minutes. To change the timeout interval, change <Server><Session><TimeOutInMin> tag in the PowerEditorConfiguration.xml file.

<TimeOutInMin>10</TimeOutInMin>

### Single-Sign-On Support

PowerEditor supports integration with Single-Sign-On (SSO) technologies, such as CA SiteMinder. It makes use of the id of an authenticated user in cookies or HTTP headers in a request made to the PowerEditor launch page. The SSO Integration feature includes:

* PE will use the authenticated user id either from HTTP header of Cookie from the request made to the launch page (http://<server>:<port>/powereditor/auth/launch.jsp). The specific HTTP header name and cookie name is specified in the PE configuration file. If no such cookie or HTTP header is found, the request will be redirected to the login page.
* PE can be configured to use an external login page, such as SSO login page, in place of the standard PowerEditor login page (http://<server>:<port>/powereditor/login.jsp). If an external login page is specified in PE configugration file and a user browses to the standard login page, the request will be redirected to the configured external login page.
* PE can be configured to use an external logout page, such as SSO logout page. This allows users logout from PowerEditor to properly dispose any session information a SSO technology maintains. If no external logout page is configured, a standard PE logout screen will be shown. If desired, PE can read the logout URL from the HTTP header.

Note that PE only supports SSO authentication, not authorization. Authorization of specific resources in PE will be validated using existing PE privilege-role authorization mechanism. Hence, in order for SSO integration to properly work, user details must be maintained in PE database just as before, with the exception of users’ password.

This section describes how to configure SSO integration and provides a typical example of use.

To configure SSO integration, the following elements should be added to the <Server> element within the PowerEditorConfiguration.xml:

<Session>

<UserIDCookie>Cookie Name</UserIDCookie>

<LoginUrl>Login URL</LoginUrl>

<Logouturl>Logout URL</LogoutUrl>

<LogoutHttpHeader>HTTP Header that contains Logout URL</LogoutHttpHeader>

</Session>

The following table summaries the SSO configuration parameters.

|  |  |
| --- | --- |
| **Name** | **Description** |
| UserIDCookie | The name of the cookie or HTTP header that contains the authenticated user id. The value of this cookie or HTTP header must be set to a clear text. If a request to the PE launch page doesn’t have any value for this cookie or HTTP header, the request will be redirected to the login page. |
| LoginUrl | The URL of the SSO login page. Requests to the PE standard login page or unauthenticated requests to the launch page will be redirected to here. |
| LogoutUrl | The URL of the SSO logout page. PE will make a request to this URL upon log out. (Note: PE doesn’t have a logout button per se. Exiting the PE applet triggers the logout action). |
| LogoutHttpHeader | The name of the HTTP Header that contains the SSO Logout URL. If specified, PE will extract the logout URL specified in the HTTP Header of a request to the launch page. |

**Configuration of SSO Software**

This section briefly discusses a recommended approach to the configuration of SSO software for PowerEditor.

1. Protect HTTP requests to all resources with a path that begins with /powereditor/auth. That is, SSO software should require authentication before it grants access to all resources under http://<server>:<port>/powereditor/auth/.

2. Configure SSO to set the authenticated user id either in a cookie or pass it in the HTTP header, as part of a request to the PE launch page.

3. PowerEditor will redirect users to the configured SSO logout page when users exist PE application. Typically, this will terminate the user’s SSO session. If this is not desired, do not configure the Logout URL (leave it blank or comment out).

With SSO configured, users should access PowerEditor by browsing to the following:

http://<server>:<port>/powereditor/auth/launch.jsp

## Enumeration Source Configuration

PowerEditor supports XML files as a source of enumeration values for guideline template columns. Each enumeration source is to be configured in PowerEditorConfiguration.xml. This allows multiple templates to share the same enumeration source. This section describes how to configure enumeration source and provides a typical example of use. Configuration enumeration source is optional.

### Enumeration Source Declaration

Enumeration source sare declared in PowerEditorConfiguration.xml using <EnumerationSource> element in <EnumerationSources> element inside the <Server> element. For example, add to PowerEditorConfiguration.xml the following:

<server>

...

**<EnumerationSources>**

**<EnumerationSource>**

**<Type>XML</Type>**

**<Name>States</Name>**

**<SupportsSelector>false</SupportsSelector>**

**<Param name="xml-file" value="state-enumeration.xml"/>**

**</EnumerationSource>**

**<EnumerationSource>**

**<Type>XML</Type>**

**<Name>Counties</Name>**

**<SupportsSelector>true</SupportsSelector>**

**<Param name="xml-file" value="county-enumeration.xml"/>**

**</EnumerationSource>**

**</EnumerationSources>**

</server>

The 'state-enumeration.xml' and 'county-enumeration.xml' files referenced above are provided in the PowerEditor distribution file.

The <EnumerationSources> element contains one or more <EnumerationSource> elements. The table below describes each of configuratio elements of <EnumerationSource> elements:

|  |  |  |
| --- | --- | --- |
| EnumerationSource Tags | Description | Required? |
| Type | The type of a enumeration source. Only XML is supported. Set this to “XML.” | Yes |
| Name | The name of this enumeration source. Must be unique across all enumeration sources. | Yes |
| SupportsSelector | Indiciates whether this enumeration source supports selector. If so, the enumeration XML file set for this must have selector element set. | No. Defaults to “false”. |
| Param | For each type of enumeration source, a set of Param elements are required. For XML, a Param with “xml-file” must be provided. | Yes. See description. |
| Param::name | Name of a parameter. | Yes |
| Param:value | Value of a parameter. For “xml-file”, this must be set to a full path to an enumeration XML file. See Section 7.6.2 for details. | Yes |

Figure 6‑ Enumeration Source Configuration Elements

### Format of Enumeration XML Files

The specified XML must conform to the following XML Schema. This schema file, XMLEnumerationFile.xsd, is included in the PowerEditor distribution file under 'config' directory.

<?xml version="1.0" encoding="utf-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

<xsd:element name="PowerEditorEnumeration">

<xsd:complexType>

<xsd:sequence>

<xsd:element name="EnumValue" type="EnumValueType" minOccurs="1" maxOccurs="unbounded" />

</xsd:sequence>

</xsd:complexType>

</xsd:element>

<xsd:complexType name="EnumValueType">

<xsd:sequence>

<!-- Value stored in PE KB and used in generated rules. -->

<xsd:element name="Value" type="xsd:string" minOccurs="1" maxOccurs="1" />

<!-- Value displayed in UI for user selection. -->

<xsd:element name="DisplayLabel" type="xsd:string" minOccurs="0" />

<!-- Indicates if this enum is inactive. -->

<xsd:element name="Inactive" type="xsd:boolean" minOccurs="0" />

<!-- Selector value for this enuemration value, if selector is supported. -->

<xsd:element name="SelectorValue" type="xsd:string" minOccurs="0" />

</xsd:sequence>

</xsd:complexType>

</xsd:schema>

### Sample Enumeration XML File

Here is a sample enumeration XML. Sample enumeration XML files are provided in the 'config' directory in the PowerEditor distribution file.

<!-- State -->

<PowerEditorEnumeration>

<EnumValue>

<Value>GA</Value>

<DisplayLabel>Georgia</DisplayLabel>

</EnumValue>

<EnumValue>

<Value>VA</Value>

<DisplayLabel>Virginia</DisplayLabel>

</EnumValue>

</PowerEditorEnumeration>

<!-- County -->

<PowerEditorEnumeration>

<EnumValue>

<Value>forsyth</Value>

<DisplayLabel>Forsyth</DisplayLabel>

<SelectorValue>GA</SelectorValue>

</EnumValue>

<EnumValue>

<Value>fulton</Value>

<DisplayLabel>Fulton</DisplayLabel>

<SelectorValue>GA</SelectorValue>

<Inactive>true</Inactive>

</EnumValue>

</PowerEditorEnumeration>

### Example of Use

Here is an example of what each project needs to do support Geographic Data columns.

1. Configure State Enumeration Source in PowerEditorConfiguration.xml.
2. Configure County Enumeration Source in PowerEditorConfiguration.xml.
3. Configure addition sources if required, such as zip codes and city.
4. For each template that needs country, add a state column and a county column of type EnumList.
5. Select State Enumeration Source for the state column. For the county column, select County Enumeration Source as an enumeration source and State column as the selector column (See Figure below).
6. Complete Template rule by using the state and county column as you would use any other EnumList column.

## LDAP, Authentication, and Authorization

### Overview and Definitions

PowerEditor provides the capability to store and retrieve user authentication and authorization information, either internally, externally, or both. For those not savvy with the lingo:

*Authentication:* Verify that the username/password is an authentic login for the PowerEditor

*Authorization:*  Verify what actions authenticated user is authorized to perform.

*LDAP:* Light-weight Directory Access Protocol – This protocol is used for accessing hierarchical views of various objects, and is commonly used for retrieving and storing authentication/authorization information.

*Privileges:* A list of various actions that users might be authorized to perform. PowerEditor privileges include actions like View Entity Data, Edit Entity Data, Manage Guidelines, Export Data, and Deploy. Privileges are mostly hard-coded in the PowerEditor, since they are very application specific.

*Roles:* An aggregation of privileges that define a *set* of actions required for a person to fulfill a business role. For example, a business analyst role would have privileges for Entity Data, Edit Entity Data, Manage Guidelines, while a rule administrator role would have Export Data and Deploy privileges. Roles are very site specific, and are customizable in the PowerEditor.

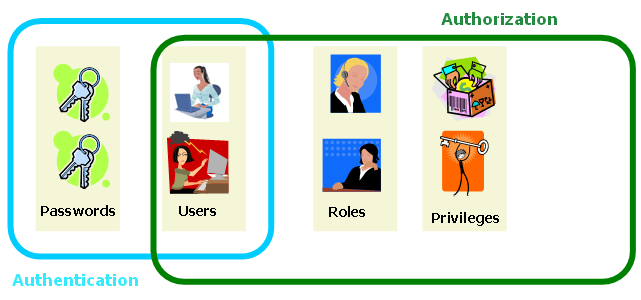


Figure 6‑ Definition of Authentication and Authorization

PowerEditor provides the ability to configure authentication and authorization for the following 3 scenarios:

* PE Authentication / PE Authorization (shortened as *PE Authentication*)
* LDAP Authentication / LDAP Authorization (shortened as *LDAP Authentication*)
* Custom Authentication / PE Authorization (shortened as *Custom Authentication*)

##### PE Authentication

Everything is stored in the PowerEditor DB: there are no external interfaces.

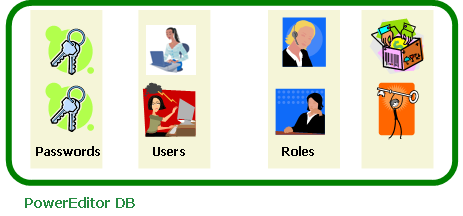


Figure 6‑ PE Authentication/Authorization

##### LDAP Authentication

Users, Passwords and Roles are stored externally, and accessed via an LDAP API. Privileges are stored in PowerEditor DB. In this case, roles are defined in the PowerEditor, and are referenced in the LDAP tables.

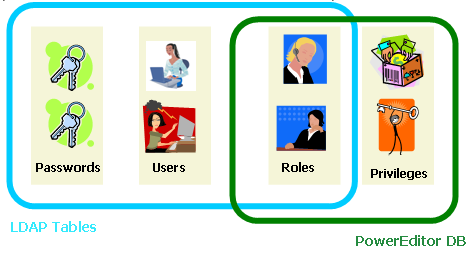


Figure 6‑ LDAP Authentication/Authorization

##### Custom Authentication

Everything is stored in the PowerEditor database except passwords. At login time, PowerEditor passes username and password to a custom plug-in (e.g. an EPASS plug-in), which authenticates the login. In this scenario, user names need to be stored in the PowerEditor database so that they can be mapped to PE-defined roles. However, the main authentication point for users and passwords is in the external system.

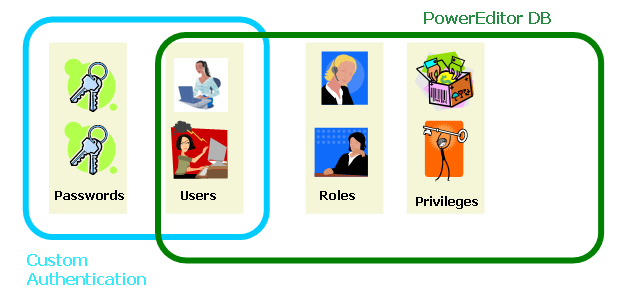


Figure 6‑ Custom Authentication / PE Authorization

### PE Authentication Configuration Overview

The PowerEditor Configuration file has 3 main elements pertaining to authorization and authentication. The following is an overview of how to configure these elements for the above 3 scenarios.

##### PE Authentication and Authorization

To configure the PowerEditor to use PE authentication and authorization:

1. Ignore the <Server><LDAP> element (it doesn’t matter if this is commented out or not)

2. Comment out the <Server><Database><UserManagementProviderClass> element, disabling LDAP authorization

3. Comment out the <UserAuthenticationClass> element, disabling any external authentication

##### LDAP Authentication and Authorization

To configure the PowerEditor to use LDAP Authentication and Authorization:

1. Configure the <Server><LDAP> elements, as described below in the section titled PowerEditor LDAP Configuration Options*.*

2. Uncomment the following element, thus enabling LDAP authorization

<Server>

<Database>

<UserManagementProviderClass>com.mindbox.pe.server.db.LDAPUserManagementProvider

</UserManagementProviderClass>

3. Uncomment the following element, the enabling LDAP authentication

<Server>

<Session>

<UserAuthenticationClass>com.mindbox.pe.server.spi.db.DefaultUserAuthenticationProvider

</UserAuthenticationClass>

##### Custom Authentication / PE Authorization

To configure the PowerEditor to use Custom Authentication and PE Authorization:

1. Ignore the <Server><LDAP> elements (it doesn’t matter if they are commented out)

2. Comment out the <Server><Database><UserManagementProviderClass> element, thus disabling LDAP authorization.

3. Uncomment and edit the following element, thus enabling a custom authentication. In this element, specify the name of the class that provides an interface to the custom authentication server. Use the fully qualified classname for the custom class, and put the class file in the server classpath.

<Server>

<Session>

<UserAuthenticationClass>com.xyz.powereditor.security.CustomAuthenticator

</UserAuthenticationClass>

### LDAP Overview

Now that we have a good understanding of authentication and authorization, let’s talk specifically about LDAP. This is a brief overview of LDAP as it pertains to the PowerEditor.

LDAP (Light weight Directory Access Protocol) is an API for communicating with directory server applications, *LDAP servers*, which provide hierarchical view of various objects. Unlike DBMS, LDAP servers maintain a hierarchy of objects. Each object in LDAP servers has attributes, which are analogous to columns of a DB table. But, LDAP object attributes differ from DB table columns in that (1) unused attributes do not take up any physical disk space and (2) attributes may contain more than one value.

To use LDAP for authentication or authorization, the PowerEditor will need be able to access a preinstalled LDAP server. The LDAP server must support LDAP v3. Most customers that require LDAP will already have an LDAP server preinstalled. However, for testing, you might want to install an LDAP server and client on your machine. *Appendix I: LDAP Server Sample Installation* contains instructions for installing OpenLDAP, and a very lightweight LDAP client called JXplorer. These applications are not intended for production purposes.

An LDAP server has user objects and container objects (meaning that they can have other objects in them). User objects typically have user id and password attributes, and are typically contained within a container object. Each LDAP object has an attribute called *objectclass* which identifies what kind of object it is. User objects will typically have *inetOrgPerson* or *user* as a value of their *objectclass* attribute. In most cases, you will leverage these existing user nodes. However, for some degenerate cases, PowerEditor provides a custom user *objectclass* called *powerEditorOrgPerson*.

##### PowerEditor LDAP Configuration Options

To configure LDAP for authentication and authorization, you most perform the three steps itemized in the above as outlined in the LDAP Authentication and Authorization section.

<LDAP>

<Connection>ldap://localhost:389</Connection>

<AuthenticationScheme>simple</AuthenticationScheme>

<Principal>cn=Manager, dc=mindbox, dc=com</Principal>

<Credentials>hkQ0vmOMSYI=</Credentials>

<UserDirectoryDN>ou=Users, o=PowerEditor, dc=mindbox, dc=com</UserDirectoryDN>

<UserObjectClassHierarchy>top,person,organizationalPerson,powerEditorOrgPerson

</UserObjectClassHierarchy>

<UserObjectRequiredAttribute name="sn" value="name"/>

<UserIDAttribute>cn</UserIDAttribute>

<UserPasswordAttribute>userPassword</UserPasswordAttribute>

<UserNameAttribute>cn</UserNameAttribute>

<UserStatusAttribute>peUserStatus</UserStatusAttribute>

<UserRolesAttribute>peRoleName</UserRolesAttribute>

<UserPasswordChangeRequiredAttribute>pePasswordChangeRequired</UserPasswordChangeRequiredAttribute>

<UserPasswordChangeDateAttribute>pePasswordChangeDate</UserPasswordChangeDateAttribute>

<UserFailedLoginCounterAttribute>peFailedLoginCounter</UserFailedLoginCounterAttribute>

</LDAP>

The table below describes each of these elements:

|  |  |  |
| --- | --- | --- |
| LDAP Tags | Description | Req in Access Only Mode? |
| Connection | The LDAP connection string specifies the location of LDAP server.  This can be a Distinguished Name (DN). Examples: ldap://localhost:389 or ldap://mbhqpdc.mindbox.com:389 | Yes |
| AuthenticationScheme | Authentication method for connecting to the LDAP server. Valid values are anonymous and simple. To connect anonymously (w/o userid and pwd) use anonymous. Otherwise, use simple. Note that the following options are not supported: DIGEST-MD5, MD5, CRAM-MD5, GSSAPI, EXTERNAL | Yes |
| Principal | Principal user id for connecting to the LDAP server. | Required only if AuthenticationScheme is not set to anonymous. |
| Credentials | Credentials (password) used for connecting to the LDAP server. This must be an encrypted password. If this password is not used, but is not commented out, it still must be encrypted[[2]](#footnote-2). | Required only if AuthenticationScheme is not set to anonymous. |
| UserDirectoryDN | Specify the Distinguished Name (DN) of directory that contains user objects. If you need to specify more than one DN, simply add more UserDirectoryDN elements.  If no UserDirectoryDN is specified, PE will use the initial directory. In this case, Connection string should have DN in it. | No |
| UserObjectClass | Name of user object’s *objectclass* attribute. This is optional. It defaults to person (inetOrgPerson) | No |
| AllowUpdate | Deprecated in PE 5.2 – updates no longer supported |  |
| AlllowChangePassword | Deprecated in PE 5.2 – updates no longer supported |  |
| AllowHardDelete | Deprecated in PE 5.2 – updates no longer supported |  |
| UserObjectClassHierarchy | Deprecated in PE 5.2 – only used for updates which are no longer supported |  |
| UserObjectRequiredAttribute | Deprecated in PE 5.2 – only used for updates which are no longer supported |  |
| UserIDAttribute | Specifies name of LDAP user ID attribute that gets mapped to PE user ID property. Typical values include *cn* and *uid* (Using these values is recommended) | Yes |
| UserPasswordAttribute | Specifies name of LDAP password attribute that gets mapped to PE password property. Defaults to *userPassword*. Note: Do not be misled into thinking this a password that needs to be encrypted: this is the name of the password *attribute*. | No |
| UserNameAttribute | Specifies name of LDAP user name attribute that gets mapped to PE user name property. If UserNameAttribute is not specified, it is set to the *cn* of the LDAP user object – in which case the user's name cannot be changed in PE GUI. |  |
| UserStatusAttribute | Specifies name of LDAP user status attribute that gets mapped to PE user status property. If not specified, all user status will be set to Active, and user’s status cannot be changed in the PE GUI. Otherwise, PE will only treat the user as Active of LDAP table entry is *Active.* | No |
| UserRolesAttribute | Attribute of user objects that holds roles of a user. This must refer to a multi-value attribute. If not specified, user roles will be maintained in PE database, and you will be able to modify roles using PowerEditor | No |

Figure 6‑ LDAP Configuration Elements

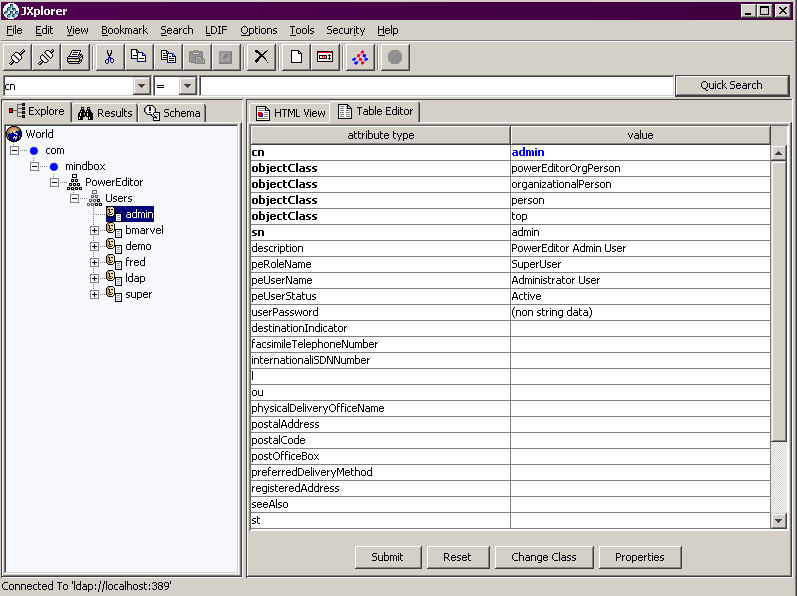


Figure 6‑ Sample Client View of LDAP User Class

## Password Policies

### Overview

PowerEditor can be configured to enforce site specific password policies such as minimum length, minimum number of special characters, and expiration periods. This section describes these features.

PowerEditor supports the following password policy enforcement:

* String Validation
  + minimum password length (e.g. must be at least 7 characters)
  + minimum pattern matches (e.g. 1 numeric and 1 upper case). These pattern matches are specified via regular expressions.
* Password Expiration
  + the number of days it takes for a password to expire
  + the number days *prior* to expiration that the user will receive notification
* Lockout after repeated login failures
  + The number of failed attempts it takes for a user to be automatically locked out
* Password History
  + The user is prevented from reusing a number of historical passwords

Note that it does not make sense to use the PowerEditor password policies if authentication is performed by an external application (e.g. LDAP). If both password policies and external authentication are enabled, password polices are ignored.

### Example Password Policies

Let's say that you would like to enforce the following password policies:

Passwords must be a minimum of six characters, must not equal the previous three past passwords, and must contain at least three of the following four criteria: One upper case alpha character, one lower case alpha character, one numeric character, one non-alphanumeric character.

Here is a configuration file excerpt that implements the above password policies:

<UserPasswordPolicies>

<Validator providerClassName="com.mindbox.pe.server.spi.pwd.RegexpPasswordValidator">

<Param name="minLength" value="6"/>

<Param name="minRegexpMatch" value="3"/>

<Param name="regexp" value=".\*([\p{Upper}]).\*"/>

<Param name="regexp" value=".\*([\p{Lower}]).\*"/>

<Param name="regexp" value=".\*([\p{Digit}]).\*"/>

<Param name="regexp" value=".\*([\p{Punct}]).\*"/>

<Param name="description" value="Passwords must be a minimum of six characters, must not equal the previous three past passwords, and must contain at least three of the following four criteria: One upper case alpha character, one lower case alpha character, one numeric character, one non-alphanumeric character"/>

</Validator>

<Expiration expirationDays="90" notificationDays="14"/>

<Lockout maxAttempts="3"/>

<History lookback="3"/>

</UserPasswordPolicies>

### Lockout Behavior

The following describes how users can be locked out, and how they can become unlocked. In this discussion, a *system administrator* is defined as someone who has the "Manage User" privilege enabled.

A PowerEditor user can be locked out for one of 3 reasons:

1. Admin user manually changed the user's status to "Locked out"
2. User exceeded failed login attempts threshold
3. User failed to change his password before it expired.

Once a user has been locked out, when the attempt to login they will receive the same message independent of the reason for lockout.



Figure 6‑ - User Locked Out!

Once a use has been locked out, an administrator must intervene by either changing the user's status to "Active" for case 1 above, or by resetting the user's password for cases 2 and 3.

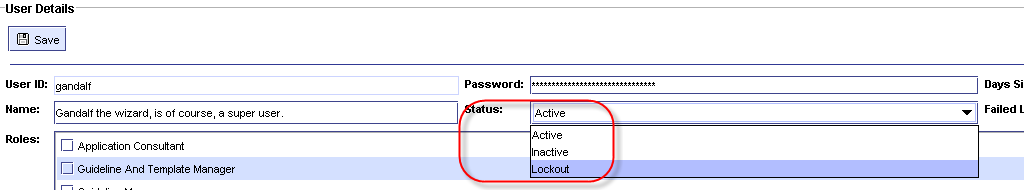


Figure 6‑ – Change Status to Active if Manually Locked Out

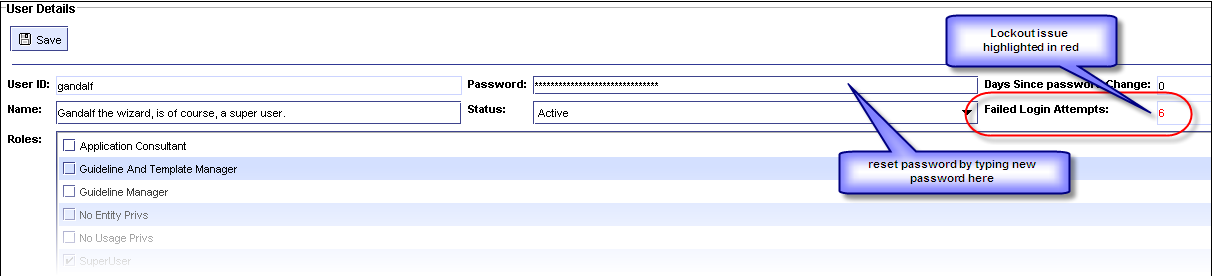


Figure 7‑ – Reset User Password if Expired or Exceed MaxAttempts

### Change Password Restriction

Users cannot change their own password for a certain period of time since last change. This time interval is configurable. If desired, this restriction can be turned off via configuration file. Use the cannotChangeIntervalMins attribute of the <Change> element in the <UserPasswordPolicies> element in the <Server> section, as follows:

<PowerEditorConfiguration>

...

<Server>

...

<UserPasswordPolicies>

**<Change cannotChangeIntervalMins=’60’/>**

Note that setting the value to a non-positive value turns off time-based password change restriction.

### Password Policy Configuration File Specifications

The following details password policy configuration options. Note that if no password validation configuration options are specified, then passwords never expire, can be set to any non-empty string. Also, the user is never locked out except by manual intervention by the system administrator.

|  |  |
| --- | --- |
| UserPasswordPolicy Tags | Description |
| **Validator** | This section contains the specification for valid password strings. If this section is absent, there are no restrictions on the password string. |
| Validator::**providerClassName** | This is a Java plugin for the code that handles password validation. The below validator parameters are used by the default validation plug-in, RegexpPasswordValidator |
| Validator::Param[**name**] | must be one of minLength, minRegexpMatch, regexp, or description |
| Validator::Param[name=**minLength**] | The minimum length of the password. **value**= is required, and must be an integer that specifies the minimum length. If the validator sees no minLength parameter, then no minimum length is enforced. |
| Validator::Param[name=**minRegexpMatch**] | The number of times that the password must match the specified regular expressions. **value**= is required and must be an integer which specifies the number of regular expressions that match. If single regexp matches twice, it only counts once toward the minRegexpMatch. If the minRegexpMatch parameter is missing, or the value is 0, all regular expressions are ignored. |
| Validator::Param[name=**regexp**] | A regular expression that specifies a pattern that must appear in the password. **value=** is required and must be a string specifying a Java regular expression. |
| Validator::Param[name=**description**] | If the user enters a new password, and the password fails to match the validator criteria, this string is displayed to the user. This string should be a textual description of all the criteria, including History and Validator parameters. Note that expiration and lockout parameters do not need to be included here, since the PowerEditor displays a different string. **value=** is required and must be a string. |
| **Change** | This section contains rules for password change. If this section is absent, passwords change restriction will not be enforced. |
| Change[**cannotChangeIntervalMins**] | Indicates time interval in which password changes is disallowed, in minutes. |
| **Expiration** | This section contains rules for when passwords expire. If this section is absent, passwords never expire. If a password expires before the user changes it, the user account becomes locked, and requires a system administrator to unlock. The user will simply receive a message saying that their account is locked. |
| Expiration[**expirationDays**] | The number of days it takes for a password to expire. This is measured from the date the user is created, or from the last time the password was reset. |
| Expiration[**notificationDays**] | The number of days prior to expiration that the user receives notification that their password is about to expire. This notification occurs on login. |
| **Lockout** | This section specifies the number of failed attempts it takes for a user to be automatically locked out. Once locked, a system administrator needs to unlock the account. |
| Lockout[**maxAttempts**] | Specifies the number of attempts prior to lockout. |
| Lockout[**resetIntervalMins**] | Specifies time interval after which login failure count is reset to zero, in minutes. |
| **History** | This section is used to restrict valid passwords based on a user's historical passwords. |
| History[**lookback**] | The number of historical passwords that are considered invalid. For example, if this is 2, then the user's previous 2 passwords would be considered invalid. If the user attempts to set their password to either of the two passwords, they would receive the message specified in Validator::Param[description]. It is important to note that PE messaging does not differ if a password is invalid due to history, length, regexp, etc. The description message needs to cover all these cases. |
|  |  |

Figure 6‑ Password Policy Configuration Elements

## Summary of Configuration Changes for Backward Compatibility

PowerEditor 5.9.0 introduced changes in rule generation behavior. This section summarizes the list of changes needed to ensure that PowerEditor generates the same set of rules as previous versions, such as 5.8.3.

* Add <TimeOutInMin>30</TimeOutInMin> in the <Session> section in the <Server> section (See Section 6.7.6.2)
* Place <ObjectGenerationDefault> as a direct child of <RuleGeneration> (See Section 6.6.7)
* Add <MergeRuleFilesByUsageType>true</MergeRuleFilesByUsageType to <RuleGeneration> (See Section 6.6.2)
* Add <IgnorePrecision>false</IgnorePrecision> to <RuleGeneration> (See Section 6.6.3)

# Advanced Topics

## Support for Http-Only Cookie

PowerEditor contains support for Http-Only Cookies for added security. Enabling Http-Only cookies is application container-dependent. Consult container documentation for details.

No changes to PowerEditor configuration are necessary to support Http-Only cookies.

## Notify SSO Authentication on Successful SSO Launch

PowerEditor can be launched using SSO (Single-Sign-On). PE provides a way to notify external systems when a user signs into PowerEditor using SSO. This notification was not possible in previous versions.

The actual notification mechanism differs per client installation and may require custom development.

## User Tracking Table Support

PowerEditor has an optional feature that maintains a user-tracking table in the database. Some clients use user-tracking table to apply various security-related rules to users. For instance, PowerEditor can keep track of user’s last login time. A client’s external tool would periodically scan the user-tracking table and disable or lock out users how have not logged into the application for a certain period of time. Now, when adding a new user, PE sets last login date of the new user to a past date. This allows clients user-tracking table scanning tool to handle new users properly.

Note that this feature requires custom development, as the exact schema of the user-tracking table would differ per client. A client-specific extension module is needed to take advantage of this feature.

# Configuration File Reference

This section contains a quick, at a glance reference for the configuration elements. PowerEditor provides a XML Schema file named 'PowerEditorConfiguration.xsd' in the <install-dir>/xsd.

In the tables below, the Tags column contains specification for XML tags in the following format:

*element-name***[***attribute-name***]**

*element-name***::***element-name*

*element-name***::***element-name***[***attribute-name***]**

For example, in the EntityConfig table, there is the following entry:

EntityType::EntityProperty[displayName]

This specifies the following in XML:

<EntityConfig>

<EntityType name="product" ..../>

<EntityProperty name="code" **displayName**="Product Code" .../>

## Summary of Knowledge Base Filter Parameters

Knowledge Base Filter parameters allow filtering the entire PE Knowledge Base (KB) based on a date range. See Section 6.1 KB Filtering With Date Range for more information about these configuration parameters.

|  |  |  |  |
| --- | --- | --- | --- |
| KnowledgeBaseFilter Tags | Description | Valid Values | Default Value |
| KnowledgeBaseFilter::DateFilter | Enable date-ranged based filtering |  |  |
| KnowledgeBaseFilter::DateFilter[BeginDate] | Specify the begin date. | Date in yyyy-MM-dd’T’HH:mm:ss format. | None |
| KnowledgeBaseFilter::DateFilter[EndDate] | Specify the end date. This is optional. If this is specified, PE operates in read-only mode. | Date in yyyy-MM-dd’T’HH:mm:ss format. | None |
|  |  |  |  |

## Summary of Feature Configuration Parameters

Feature configuration parameters specify which high level tabs are to be turned on and off. High level tabs that can be configured include the CBR, Phases, and Parameters tabs. See *Section 6.1 High-Level Tab Configuration* for more information about these configuration parameters.

The following are tags that appear under the **FeatureConfig** element. Note that if a feature is not present in the config file, the feature will be enabled by default.

|  |  |  |  |
| --- | --- | --- | --- |
| FeatureConfig Tags | Description | Valid Values | Default Value |
| Feature | Enable or disable a specific PowerEditor Tab |  |  |
| Feature[name] | Feature to enable or disable | One of cbr, parameter, or phase | no default |
| Feature[enable] | A boolean that specifies if the given tab is enabled | true, false | no default |
| Feature[className] | Name of class that holds the tab | see values in diagram below |  |

<FeatureConfig>

<Feature name="**cbr**" enable="false" className="com.mindbox.pe.client.applet.cbr.CBRPanel"/>

<Feature name="**parameter**" enable="true"

className="com.mindbox.pe.client.applet.parameters.ParameterManagerTab"/>

<Feature name="**phase**" enable="true"

className="com.mindbox.pe.client.applet.admin.ProcessManagementTab"/>

</FeatureConfig>

## Summary of Entity Configuration Parameters

Entity Configuration Parameters configure management of entities.

The following are tags which appear under the **EntityConfig** element:

|  |  |  |
| --- | --- | --- |
| EntityConfig Tags | Description | Valid Values |
| CategoryType | Creates category types to be associated with a generic entity |  |
| CategoryType[name] | The display name of the category (e.g. Program Category) | A string |
| CategoryType[typeID] | A type ID to be referenced in the generic entity definition (see below) | An integer |
| EntityType[name] | The internal name of the entity type. This serves as an identifier for the entity type. The value of this cannot be *id* or *name*, since these are predefined attributes automatically associated with each entity type. | A symbol |
| EntityType[displayName] | Name displayed for the entity. | A string |
| EntityType[typeID] | This needs to be a unique identifier. | An integer |
| EntityType[useInContext] | Specifies whether this entity get displayed in the guideline context selection. | Yes, No |
| EntityType[useInCompatibility] | Specifies whether this entity get displayed in the compatibility matrix. | Yes, No |
| EntityType[canClone] | Specifies whether the clone button is enabled for this entity. Cloning an entity puts the new entity in the context of all the guidelines that have the old entity. | Yes, No |
| EntityType[categoryType] | The typeID associated with the categories for this entity. This must match a value in CategoryType[typeID] | An integer |
| EntityType[useInMessageContext] | Specifies whether this entity is displayed in the message creation context. Only one entity can have this attribute set to yes. | Yes, No |
| EntityType[uniqueCategoryNames] | Specifies whether duplicate node names are allowed in the category tree for this entity type | Yes, No |
| EntityType::EntityProperty | Each occurrence of an EntityProperty defines a new property that will be displayed in association with this generic entity. |  |
| EntityType::EntityProperty[name] | The internal name of the property | A string |
| EntityType::EntityProperty[displayName] | The name shown in the GUI for the property | A symbol |
| EntityType::EntityProperty[showInSelectionTable] | Specifies whether this property is shown in the table which summarizes all instances of this entity. This table appears in the upper section of the entity definition screen. | Yes, No |
| EntityType::EntityProperty[isRequired] | Specifies whether the user is required to enter a value for this. | Yes, No |
| EntityType::EntityProperty[isSearchable] | Specifies whether this attribute appears in the filter screen | Yes, No |
| EntityType::EntityProperty[type] | The type of property that is being defined. This declares both the data type and data display. | one of: enum, string, boolean, symbol, integer, long, currency, percent, float, double, date |
| EntityType::EntityProperty[typeEnum] | Valid only when type=enum. Also required if type=enum. Specifies a database table in which enumerated values are listed. | A symbol |
| EntityType::EntityProperty[attributeMap] | Valid only when type=enum and when enumType is not used. This specifies a pointer to an enumerated list that is in the domain file. For example,  attributeMap=”LDO\_LOAN\_FEATURES.LoanDocumentationType” | an object.attribute reference |
| EntityType::EntityProperty[sort] | Valid only when type=enum. Specifies whether the PE should sort the enumerated values. | Yes, No |
| EntityType::EntityProperty [autoUpdatedDateProperty] | Valid only when type=date. Specifies that this date will be automatically updated whenever the specified property changes. | A symbol (property name) |
| EntityType::EntityProperty[allowMultiple] | Valid only when type=enum. Specifies whether the property allows multiple selections. | Yes, No |

## Summary of UI Configuration Parameters

|  |  |  |
| --- | --- | --- |
| UserInterface Tags | Description | Valid Values |
| UserInterface::Entity | Entity UI specification | n/a |
| UserInterface::Entity[showTab] | Show or hide Entity management UI altogether | Yes, No |
| UserInterface::Entity::EntityTab | Entity Tab specification | n/a |
| UserInterface::Entity::EntityTab[type] | The Entity Type represented by this tab. | must be the internal name of an entity type as defined in  EntityConfig::EntityType[name] |
| UserInterface::Entity::EntityTab[showTab] | Show or hide particular entity type tab (e.g. Product) | Yes, No |
| UserInterface::Entity::EntityTab::EntityPropertyTab | Define a new tab to group properties within an entity. | n/a |
| UserInterface::Entity::EntityTab::EntityPropertyTab[title] | The name displayed in the tab that groups properties | String |
| UserInterface::Entity::EntityTab::EntityPropertyTab::EntityPropertyName | Specify that the property should be displayed within the enclosed EntityPropertyTab | The internal name of a EntityProperty, as specified in  EntityConfig::EntityType::EntityProperty[name] |
| UserInterface::UsageTypeList | Begins declaration of usage types |  |
| UserInterface::UsageTypeList::UsageType | Begins declaration of a single usage type |  |
| UserInterface::UsageTypeList::UsageType [name] | Internal name of usageType. This is used for reference in this file, in template definition file, and is generated in rules. | Valid AE symbol (hyphens are suggested as word separation tokens – spaces are illegal.) |
| UserInterface::UsageTypeList::UsageType [displayName] | used when usageType is displayed in the PowerEditor (e.g. guideline button) |  |
| UserInterface::UsageTypeList::UsageType [privilege] | If a PE user does not have the privilege specified here, the user will not be able to view/edit guidelines of this type. | Must be a privilege defined in the PE DB |
| UserInterface::Guideline | Guideline UI specification |  |
| UserInterface::Guideline [showTemplateID] | If this is yes, show both template IDs and template names in GUI lists. Otherwise, just show template names | Yes, No |
| UserInterface::Guideline [sortEnumValue] | If this is yes, enumeration values for enumList columns are sort. | Yes, No |
| UserInterface::Guideline [fitGridToScreen] | If set to Yes, all guideline grids will be shrunk to fit the screen. If set to No, all guidelines will be displayed, with a horizontal scroll bar, as specified in the template. | Yes, No |
| UserInterface::Guideline::GuidelineTab | Guideline tab specification |  |
| UserInterface::GuidelineTab [displayName] | Shown in the GUI |  |
| UserInterface::Guideline::GuidelineTab::UsageType | Usage type specification |  |
| UserInterface::Guideline::GuidelineTab::UsageType[name] | Name of usage type. Must match with a usage type of templates in Template Definition XML | Any text |
| UserInterface::Policies:: EnforceSequentialActivationDates | Provides the option to enforce that all activation dates for a template and context are contiguous are non-overlapping. This applies to parameters and guidelines. Defaults to no. | Yes, No |
| UserInterface::DateSynonym::AllowIndenticalDates | Set to Yes to allow multiple date synonyms with the same date. Defaults to No. | Yes, No |
| UserInterface::DateSynonym::DefaultTime | When a date synonym is created, this is the default time that will be displayed. Typically, this the time at which the customer considers to be the start of a new day, in terms of their rules. | HH:MM |
| UserInterface::DeployExpirationDate::DefaultTime | Rules expired over DefaultDays ago will not be deployed, unless this value is changed in the UI. | Integer (in days) |
| UserInterface::AllowDisableEnableUser | Determines if user enable/disable functionality is used. Defaults to false. | Yes, No |
| UserInterface::ClientWindowTitle | Sets the title of the client window | String |
| UserInterface::UserDisplayNameAttribute | Determines which attribute of a user object to display as user name on screen. Defaults to ID. | ID, NAME |
| UserInterface::UnauthorizedAccessWarningText | Contains the access disclaimer verbiage displayed on the login screen. | Any text |

## Summary of Rule Generation Parameters

### Rule Generation: Overall Configuration

The following are tags which appear under **RuleGeneration**.

|  |  |  |  |
| --- | --- | --- | --- |
| RuleGeneration Tags | Description | Valid Values | Default Value |
| RuleGeneration::GuidelineMaxThread | Max number of threads used for generating guideline rules. | Positive integer | 4 |
| RuleGeneration::**IgnorePrecision** | Set to Yes to ignore template column precision setting when generating float values. | Yes, No | Yes |
| RuleGeneration::**MergeRuleFilesByUsageType** | Set to Yes to have PE generate one rule file per usage type, instead of generating one file per template. | Yes, No | Yes |

### Rule Generation: Guideline Generation Configuration

The following are tags which appear under **RuleGeneration::RuleGenerationDefault** or **RuleGeneration::RuleGenerationOverride**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RuleGeneration Tags | Description | Valid Values | Default Value | Overrideable |
| RuleGenerationOverride[usageType] | Change the rule configuration based on usage type. | Valid usageType[name] |  | Yes |
| RuleNamePrefix::Guideline | Prefix of names of ARTScript rules generated from guidelines | Valid AE symbol | GuidelineRule | Yes |
| LHS::Date | Format of date in generated rules | julian, gregorian | julian | No |
| LHS::Pattern | ARTScript pattern specification |  | no default | Yes |
| LHS::Pattern[type] | Type of pattern | request, control,  plan-evaluation, lineage, empty, ruleset | Required | Yes |
| LHS::Pattern[generate] | Turns pattern generation on or off | Yes, No | No | Yes |
| LHS::Pattern[class] | Name of class for the pattern | Valid AE class name | required | Yes |
| LHS::Pattern[prefix] | Prefix to which class name is appended | Valid AE symbol | none – no prefix prepended | Yes |
| LHS::Pattern[text] | Only for type “lineage”. Static pattern text to insert in every qualified pattern | Valid ARTScript pattern | none – no pattern inserted | Yes |
| LHS::Pattern[variable] | Variable name referring to lineage ID. Only for type “lineage”. | Valid AE variable name | ? | Yes |
| LHS::Pattern  [usageTypeAsFocus] | **Yes**=generate usageType as focus-of-attention (MMS compatible); **No**=generate GuidelineTab as focus-of-attention (pre MMS).  Used only for Pattern[type=request] | Yes, No | No (pre-MMS) | Yes |
| LHS::Pattern::Attribute | Used only for Pattern[type=control]. These items specify how the control pattern will be generated. See Section 7.4.1.2 LHS Control Pattern for full details. |  |  |  |
| LHS::Pattern::Attribute[type] | The entity type that this attribute pertains to. | Must match the name of an entity as specified in EntityType[name] |  |  |
| LHS::Pattern::Attribute[name] | The internal name of this attribute. | Must match the name of an attribute in the domain object. |  |  |
| LHS::Pattern::Attribute[value] | **Retired in PE 5.3** – PE always uses the value of the entity's ID |  |  |  |
| LHS::Pattern::Attribute[valueAsString] | **Retired in PE 4.5** – deploy type is determined by the deployType in the domain file |  |  |  |
| LHS::Pattern::Value | Specifies how to deploy patterns of a certain value |  |  |  |
| LHS::Pattern::Value[type] | The type of pattern to configure. Right now there is only one configurable value: the one used for the IS-EMPTY test | one of: unspecified |  |  |
| LHS::Pattern::Value[deployValue] | the value that gets deployed for this pattern | symbol or string |  |  |
| LHS::Pattern::Value[valueAsString] | Specifies whether the deployValue will be deployed with quotes around it. | Yes, No |  |  |
| RHS::PEActionOn | This must be Yes. | Yes |  |  |
| RHS::MessageFormatConversionFunction | function called on RHS for message formatting | symbol | sprintf | Yes |
| RHS::MessageDateFormat | Format pattern for RHS ":message" date values. This is used when a date value is known at Rules Generation time (e.g. Guideline template grid values). | Syntax is the same as for java.text.SimpleDateFormat |  |  |
| RHS::MessageDateRangeFormat | Format patterns for RHS ":message" date range values. This is used when a date range value is known at Rules Generation time (e.g. Guideline template grid values). | Syntax is the same as for java.text.SimpleDateFormat |  |  |
| RHS::MessageDateFormatAE | Format pattern for RHS ":message" date values-used when a date value is not known until the rule is evaluated by the engine (e.g. Attribute values). | Syntax is the same as for the format-julian-date ArtEnterprise function |  |  |
| RHS::MultiEnumAsSequence | Specifies how multi-enum cell values are deployed in RHS. Yes=sequence, No= comma separated string | Yes, no | Yes | Yes |
| MessageTypes | Specifies message generation defaults |  | no default | Yes |
| MessageTypes::Message | Specifies how a message should be generated for a cell |  | no default | Yes |
| MessageTypes::Message  [type] | The cell type this specification applies to. | range, enum, conditional | required | Yes |
| MessageTypes::Message[rangeStyle] | The style to use to format ranges. Applies to Message::[type=range] | **verbose**: “greater than”, etc;  **symbolic**: >, >=, etc.;  **bracketed**: “[0, 100)” | verbose | Yes |
| MessageTypes::Message[cellSelection] | For specifying text and behavior that depends on specific cell selections. Applies to Message[type=enum] | **enumExcludeSingle**: Single enum value selected, "Exclude Selection" selected;  **enumIncludeSingle**: Single enum value selected, "Exclude Selection" not selected;  **enumExcludeMultiple**: Multiple enum values selected, "Exclude Selection" selected;  **enumExcludeMultiple**: Multiple enum values selected, "Exclude Selection" not selected | required for Message::[type=enum] | Yes |
| MessageTypes::Message[enumDelimiter] | For specifying text to be used between multiple selected items;  Applies to Message[type=enum] | Any string – eg. ", " |  | Yes |
| MessageTypes::Message[enumFinalDelimiter] | For specifying text to be used between the next-to-last and last multiple selected items;  Applies to Message[type=enum] | Any string – eg. ", or " |  | Yes |
| MessageTypes::Message[enumPrefix] | Test that will be prepended to the cell value;  Applies to Message[type=enum] | Any string – eg. "Any of " |  | Yes |
| MessageTypes::Message[conditionalDelimiter] | For specifying text to be used between conditionally generated column text; Applies to Message[type=conditional] | Any string – eg. ", " |  | Yes |
| MessageTypes::Message[finalConditionalDelimiter] | For specifying text to be used between the next-to-last and last conditionally generated column text; Applies to Message[type=conditional] | Any string – eg. ", and " |  | Yes |

### Rule Generation: Object Generation Configuration

The following table appears under **RuleGeneration::ObjectGenerationDefault** tag.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object Generation Tags | Description | Valid Values | Default Value | Overrideable |
| ObjectGenerationDefault | Specifies options for parameter generation |  |  | No |
| ObjectGenerationDefault:: InstanceCreateText | AE function for creating parameter instances | One of: make-instance, create-instance |  | No |
| ObjectGenerationDefault:: ParameterContext | Used to specify the mapping between context elements and domain elements |  |  | No |
| ObjectGenerationDefault:: ParameterContext::Attribute | Specifies a particular parameter context element. |  |  | No |
| ObjectGenerationDefault:: ParameterContext::Attribute[type] | Specifies a handle to particular parameter context element. | the name of an entityType, or one of activationDate, expirationDate |  | No |
| ObjectGenerationDefault:: ParameterContext::Attribute[name] | Specifies the domain attribute to which this context element maps. | The name of a domain attribute that will be present in the AE parameter object. |  | No |
| ObjectGenerationDefault:: ParameterContext::Attribute[value] | Specifies which entity property will be used when deploying a parameter that uses an entity instance. | Must be a valid EntityProperty[name] for the given entity type. |  | No |
| ObjectGenerationDefault:: ParameterContext::Attribute[valueAsString] | Specifies whether the attribute will have quotes around it when deployed | Yes, No |  | No |

## Summary of Server Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Server Tags | Description | Valid Values | Default Value |
| KnowledgeBase::DomainFile | Location domain definition file | Valid filename |  |
| KnowledgeBase::TemplateFile | Location parameter template definition file | Valid filename |  |
| Audit::AuditAll | Turns audit trail on or off | Yes, No |  |
| Database::Provider |  | com.mindbox.pe.server.db.DefaultPEDBCProvider |  |
| Database::MaxConnection | Maximum number of concurrent DB connections allowed | Positive integer. Value >= 10 recommended |  |
| Database::MonitorInterval | ?? |  |  |
| Database::UserManagementProviderClass | Only use this element if you want to use LDAP for user authentication and authorization | com.mindbox.pe.server.db.LDAPUserManagementProvider |  |
| Database::Driver | Fully qualified name of the DB Driver class. | Fully qualified class name |  |
| Database::Connection | Database connection string. | Valid DB connection string |  |
| Database::User | Login name for accessing the DB. | Any string |  |
| Database::Password | Password for the DB user. | This must be a string that is encrypted as an output from the PowerEditor Password tool. |  |
| Database::ValidationQuery | Query execute to validate each DB connection before using it, if specified. This is an optional entry. Example: select count(\*) from MB\_PRIVILEGE | This must be a valid query select query. |  |
| EnumerationSources | For the Enumeration source entries, see Figure 6-47 Enumeration Source Configuration Elements. |  |  |
| LDAP | For the LDAP entries, see Figure 6‑53 LDAP Configuration Elements. |  |  |
| UserPasswordPolicies | For password restrictions. See Figure 6‑58 Password Policy Configuration Elements*.* |  |  |
| Deployment::BaseDir | Directory where generated ARTScript files are stored | Valid directory name |  |
| Deployment::SaveOldFiles | If Yes, Instructs PE to save previously generated ARTScript files | Yes, No |  |
| Deployment:: UseTimeStampFolder | If Yes, PE places generated ARTScript files to a time-stamp directory. | Yes, No |  |
| Deployment::ReportMissingLink | If Yes, report missing links at deployment. Use Yes here, unless you had linkage problems when upgrading from PE 3.3.0. | Yes, No | Yes |
| Deployment::PostDeployScript:File | Contains the name of an optional deploy script file. This file will be executed after rules are deployed | Valid filename | no default (no script is run) |
| Log::LogFile | XML Tag per each log file. |  | no default |
| Log::LogFile[type] | Log file type. | database, deployer, loader, server, servlet | Required |
| Log::LogFile[writeLog] | Indicate whether to write the log of the specified type. | Yes, No | No |
| Log::LogFile[debug] | Indicate whether to write debug messages. | Yes, No | No |
| Log::LogFile[file] | Full path to the log file. Make sure the directory of the file exists, or the log file won’t be written. This attribute is ignored for log type of “servlet” as Servlet log entries are written to “server” log. | Valid filename | Required |
| Log::LogFile[pattern] | Format of each log line. Must conform to Log4J format. | Valid Log4J Log entry format. | %d{yyyy-MM-dd HH:mm:sss} %c{2}: %m%n |
| Log::LogFile[maxSize] | Max size of logfile in MBytes. When file exceeds size, a new logfile will be created. | Positive Integer | No size limit |
| Session:: MaxUserSessions | Number of distinct session allowed for the same user ID | Positive integer. | 3 |
| Session::TimeOutInMin | Session timeout in minutes | Positive integer. | 30 |
| Session::UserIDCookie | The name of the cookie or HTTP header that contains the authenticated user id. The value of this cookie or HTTP header must be set to a clear text. If a request to the PE launch page doesn’t have any value for this cookie or HTTP header, the request will be redirected to the login page. | String. |  |
| Session::LoginUrl | The URL of the SSO login page. Requests to the PE standard login page or unauthenticated requests to the launch page will be redirected to here. | Valid URL. |  |
| Session::LogoutUrl | The URL of the SSO logout page. PE will make a request to this URL upon log out. (Note: PE doesn’t have a logout button per se. Exiting the PE applet triggers the logout action). | Valid URL. |  |
| Session::LogoutHttpHeader | The name of the HTTP Header that contains the SSO Logout URL. If specified, PE will extract the logout URL specified in the HTTP Header of a request to the launch page. PE will ignore LogoutUrl setting. | String. |  |
|  |  |  |  |
|  |  |  |  |
| UserAuthenticationProviderClass | Could be a custom class. If you want to use the LDAP interface for password verification, use com.mindbox.pe.server.ldap.DefaultUserAuthenticationProvider. To enable user tracking, use c om.mindbox.pe.server.sysext.usertracking.UserTrackingDbUserAuthenticationProvider | Valid class name. |  |
|  |  |  |  |

# Appendix I:LDAP Server - Sample Installation

This appendix contains instructions for installing a sample LDAP server and client for testing purposes. The applications are not intended for production purposes. The sample LDAP server documented here is OpenLDAP. The sample LDAP client documented here is a very lightweight LDAP client called JXplorer.

The server application (OpenLDAP) is what contains the user and password information. The client application is an interface that allows you to edit the information stored in the server.

Alternatively, you may use Apache Directory Server (ApacheDS) for LDAP server. It’s available at <http://directory.apache.org/apacheds/>. Also, you may use Apache Directory Studio for LDAP client, available at <https://directory.apache.org/studio/>.

## Obtain Configuration Files

Locate the following files at <install-dir>/ldap directory:

* slapd.conf
* powereditor.schema
* mindbox-dc.ldif
* powereditor-user.ldif

## Install and Configure LDAP Server Application (OpenLDAP)

1. For Windows OpenLDAP installer, download it from <http://download.bergmans.us/openldap/openldap-2.2.29/>. More recent versions might have made available since this document was written. You might find the following link to be useful: <http://lucas.bergmans.us/hacks/openldap/download>. For non-Windows installation, visit [http://www.openldap.org](http://www.openldap.org/). For any OS, you need to find a **slapd**, not *slurpd* implementation for your OS.

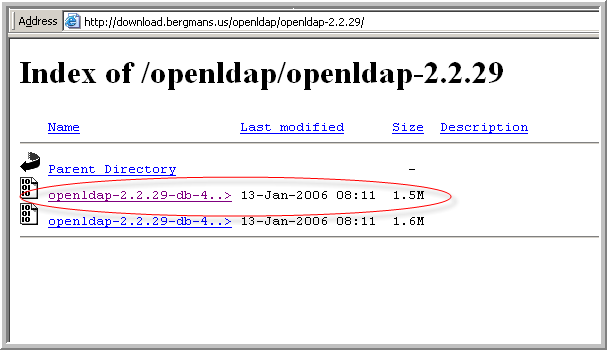


Figure 8‑ OpenLDAP Download Site

1. Run the OpenLDAP installer. Make sure to select **slapd,** not *slurpd.*

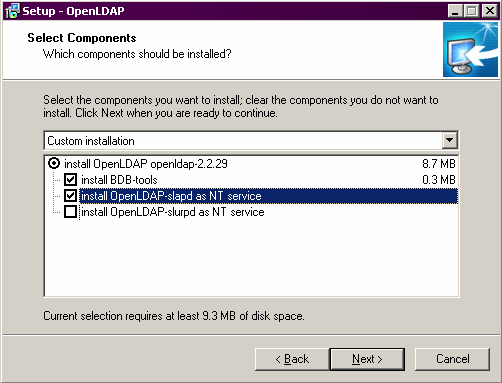


Figure 8‑ Installing OpenLDAP

1. Configure OpenLDAP with PowerEditor specific files.
   1. Place file *slapd.conf* into your <openLDAP> installation directory, overwriting the existing file.

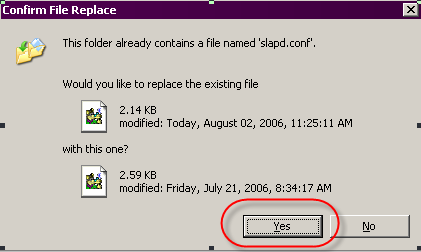


Figure 8‑ Just Say Yes

* 1. Place file *powereditor.schema* into <openLDAP>/schema directory.

1. Start OpenLDAP. It should be ready for connection.

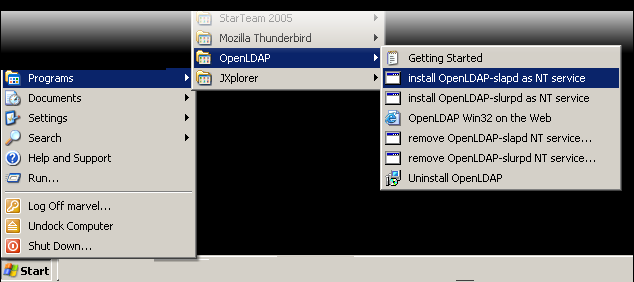


Figure 8‑ Starting OpenLDAP from Windows

1. At that point, you probably want to verify that OpenLDAP is running. Here is some background: Port 389 is typically used by LDAP servers. OpenLDAP is by default configured to listen to Port 389. To verify that this port is being used (hopefully by OpenLDAP) do the following.
   1. open a DOS window
   2. type netstat –na
   3. verify that something is listening to port 389

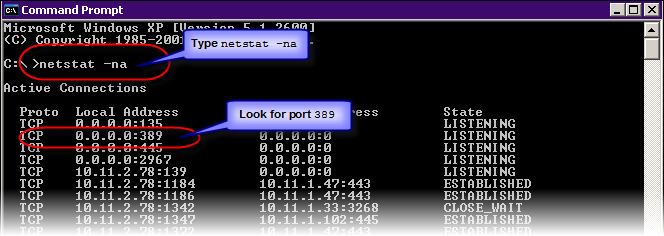


Figure 8‑ Verify that the LDAP server is running

## Install LDAP Client Application (JXplorer)

1. Install LDAP Client application. You can use JXplorer LDAP Browser, available at <http://www.jxplorer.org/> or Argonne’s LDAP Browser, available at

<http://www.novell.com/coolsolutions/tools/13765.html>.



Figure 8‑ Downloading JXplorer

1. Use your LDAP client application (e.g. JXplorer) to import the file *mindbox-dc.ldif.* Close your LDAP client application and OpenLDAP.

## Use Client Application to Configure PowerEditor LDAP Settings

1. Start your LDAP client application.
2. From your client, enter settings in PowerEditor Configuration. The UserDN and Password settings are based on values that are in the slapd.conf file.
   1. **Host**: *localhost*
   2. **Port**: *389* - This the default for LDAP servers.
   3. **Level**: *User+Password*
   4. **Password**:*secret* - This is specified in the slapd.conf file.
   5. **UserDN:** *cn=Manager, dc=mindbox, dc=com*  *-* This is specified in the slapd.conf file. Note that the PowerEditorConfig file will also contain this value in the *<Principal>* section.

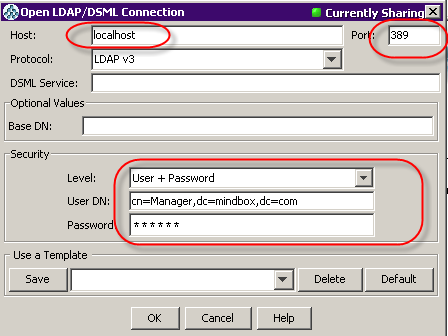
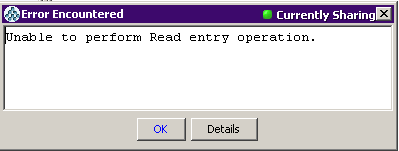
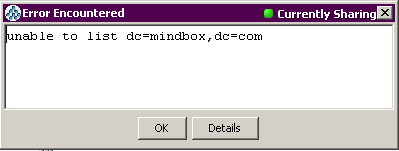


Figure 8‑ LDAP Client Application

If you get these dialogs, it’s OK.



1. Using your LDAP client application, import the file *mindbox-dc.ldif*.

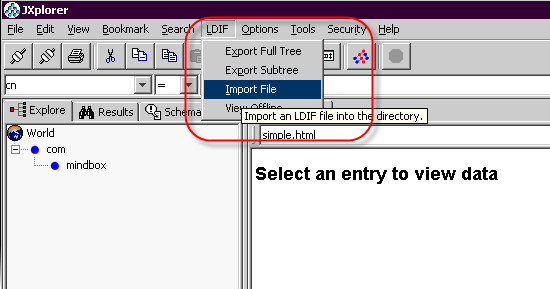
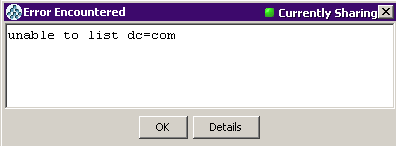


Figure 8‑ Import *mindbox-dc.ldif*

If you get this dialog, it's OK



1. Using your LDAP client application, import the file *powereditor-user.ldif*. (XXXX)

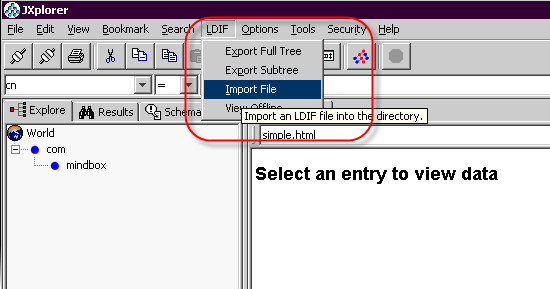


Figure 8‑ Import *powereditor-user.ldif*

If you see the PowerEditor directory structure, you've succeeded so far.

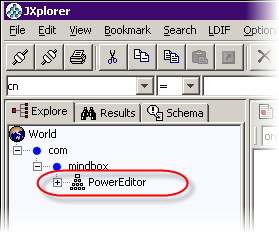


Figure 8‑ Result of Importing Two *ldif*  Files

1. Add and edit users. In JXplorer, go to the Table Editor. Use menu items to add and edit users.

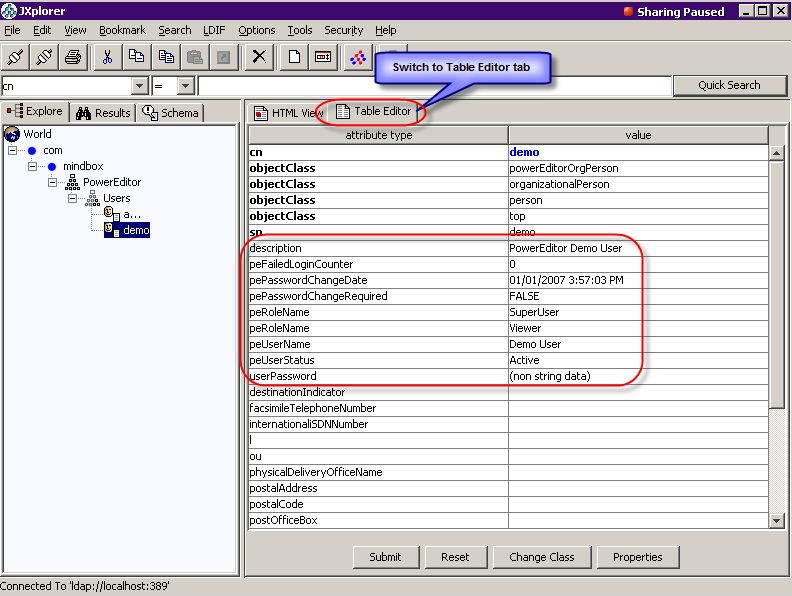


Figure 8‑ Editing LDAP User Information

If your PowerEditor is configured for LDAP, you should see the effects of these edits the next time you login to the PowerEditor.

## Restarting JXplorer

When you restart JXplorer, you’ll need to re-establish your LDAP connection.

You'll need to reenter all the data you entered in the start of the previous section, including Level, User DN and Password.

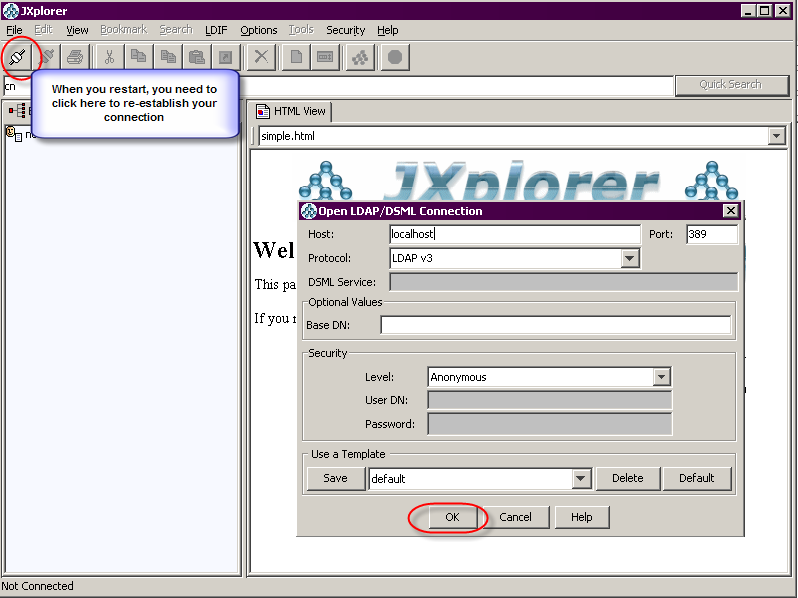


Figure 8‑ Restarting JExplorer

## Debugging Open LDAP

If the OpenLDAP service doesn't start, you can try the following

* in a DOS window, cd to the OpenLDAP directory (e.g. C:\Program Files\OpenLDAP)
* type slapd.exe
* if you see the DOS prompt immediately after hitting return, OpenLDAP didn't start properly
* type slapd.exe –d 5
* you should see a bunch of debugging information
* if it says that your database needs to be recovered, type db\_recover –h data
* now try slapd.exe
* if the DOS prompt doesn't come back, slapd is running. You can let it continue to run, or type control-C, and go start it as a service
* in a DOS window, type netstat –na to verify that port 389 is in use.

1. DomainEditor is in development at the time of this writing. Consult the Development group for more information. [↑](#footnote-ref-1)
2. See section on PowerEditor Encryption Tool [↑](#footnote-ref-2)