Expere® Engine and Tools / Expere® Authoring
2016.1

Administration Tool

User Guide

January 2016

Wolters Kluwer
Financial Services

Preface

This publication was written forExpere® Engine and Tools / Expere® Authoring

Publication Information / Version
Document Title: Administration Tool
Release Date: January 2016

Wolters Kluwer
Financial Services

Table of Contents

User GuideProvides important information about new features, enhancements, and other changes to the Expere solution in this release. Wolters Kluwer Financial ServicesWolters Kluwer Financial ServicesExpereExpereExpere

Wolters Kluwer
Financial Services

Introduction

About this guide

This guide is intended for content authors; network and database administrators; and anyone who is required to set up and administer the Expere Knowledge Base.

It describes how to work with and build content repositories, set up the precedence units, and how to analyze and decide on the impact to existing custom content when a new release of the Expere Knowledge Base has been received.

Supporting Documents

This guide is part of a set of documents that support the Expere Authoring Environment. The other guides in the documentation set are as follows:

- *Installation Guide*: describes how to install the Administration Tool and the Requirements and Content Editor, and how to configure the databases and web services.
- *Requirements and Content Editor User Guide*: describes how to author and update the content in the Requirements and Content Editor.
- *Expere Stylesheet Reference Guide*: describes the WKFS Style Guide and corresponding standard (i.e., WKFS_StyleGuide.xsl) and the override stylesheets.

Wolters Kluwer
Financial Services

Installation Prerequisites

The Content Administration Tool (Admin Tool) requires that the following components, some of which are required to work with Requirements and Content Editor (Editor) are installed.

Requirements and Content Database Server

- # SQL Server 2005 Standard Edition

Loader Database Server

- # Oracle* 10g /11g with the Expere Engine **

*For Oracle database configuration with the content loading functionality, see the Expere Implementation Guide that is included with the Expere Document Engine. **The minimum version of the Expere Engine you need depends on the lines of business you use. For example, the 2010.0 Expere Engine or newer is required for using the Deposit line of business. The Mortgage line of business requires the 2010.1 Engine or newer. Refer to the release notes that accompany new content releases to determine the version of the Engine you must use for each line of business.

Authoring Web Server

- Windows Server 2008 R2 or higher
- NET Framework 4.5
- Microsoft Internet Information Services (IIS) 7 or higher
- MS SQL Server 2008 R2 (or higher), Express, Standard or Enterprise
- 4 GB RAM, dual core processor
- Microsoft Internet Information Services (IIS) 6.0 or higher

Client

- Windows 7 or higher
- NET Framework 4.5
- Microsoft Visual Studio Isolated Shell 2013
- MS SQL Client Tools for loading to the Authoring Web Server or an MS SQL based Expere runtime
- 64-bit Oracle Data Access Components (ODAC) for loading to an Oracle based Expere runtime

- 16 GB RAM, quad core processor

Build Server (optional)

- Windows Server 2008 R2 or higher
- NET Framework 4.5
- Command line Expere Administration Tool
- Windows Task Scheduler for scheduled builds to run automatically
- MS SQL Client Tools for loading to the Authoring Web Server or an MS SQL based Expere runtime
- 64-bit Oracle Data Access Components (ODAC) for loading to an Oracle based Expere runtime
- 16 GB RAM, quad core processor

Wolters Kluwer
Financial Services

Admin Tool Installation

Installation

The following procedures describe how to install and set up the Admin Tool.

Database Setup

Setting up the database environment is described here at a high level. This procedure presumes that the person performing these steps will be well-versed in Microsoft SQL Server and understands the requirements for each step described.

Create the Authoring Databases

Two databases are required; one for the Requirements and Content Editor and another for the Schema Requests.

In Microsoft SQL Server, create two databases. Name one **Requirements** and the other **SchemaRequest**.

> **Note:**
> The database can be named anything, but the examples in this guide use the names listed above.

Create the Authoring Users

In the Microsoft SQL Server:

1. Create a user.
2. Add the user to the Requirements database. Set security to read/write.
3. Add the user to the SchemaRequest database. Set security to read/write.

Creating an ODBC Data Source

The Administration Tool can quickly obtain the database connection information it needs from an ODBC data source, rather than requiring the connection string to be entered manually.

Create an ODBC Data Source as follows:

1. Open the **Data Sources (ODBC)** in Windows Administrative Tools.
2. Open the **System DSN** tab and click **Add…**
3. Select the **SQL Server** driver and click **Finish**.
4. Enter the database details:
   - Name: CAEAdministrationTool (this name is not required).
   - Description: Expere Content Database.
   - Server: the SQL Server name.
5. Click **Finish**.

Connecting the Administration Tool to the Requirements and Content Database

The Admin Tool must have an Expere content repository open to connect to the Requirements and Content database. This process assumes that the repository has been loaded to your content management system or file system and is available to the Admin Tool.

After opening the Admin Tool:

1. Select **Generate Requirements & Content Editor Database** from the **Tools** menu.
2. Select **CAEAdministrationTool** from the **System Data Sources** menu.
3. Select the **User Data Source**.
4. Click **Use Below** to populate the **Build Properties**.
5. Click **Build**.

   The databases will be built and tables populated.

> **Note:**
> You can verify that the databases have been created using SQL Server Management Studio.

Configuring Schema Options

This topic describes the configuration options available for managing and defining schema files.

Overview
Associating a schema to a document is required. By default, three schema files (xsd) are supported by the solution.

- **ExpereTxn:** The base schema for the majority of documents and the default schema.
- **VMPMISMO:** The VMP Mismo schema is used primarily for documents supporting the Mortgage lending line of business.

- **QuickDoc:** The QuickDoc schema is used in conjunction with documents that are processed through the QuickDocs component to render eForms documents.

The configuration file contains records that define the available schema. The configuration file is located in the default installation path. The configuration file is located in the installation path. The default location (which can be modified) is:

```
C:\Program Files (x86)\WKFS\Expere Administration Tools\RequirementsAdmin.exe.config
C:\Program Files\WKFS\Expere Administration Tools\RequirementsAdmin.exe.config
```

In the command line version of the tool, the config file is located:

```
C:\Program Files (x86)\WKFS\Expere Administration Tools\ReqAdmin.exe.config
C:\Program Files\WKFS\Expere Administration Tools\ReqAdmin.exe.config
```

Both configuration files are located at the same installation path.

You can modify this setting to add support for custom schema files. In this example, a custom schema is defined, along with the three packaged schema files, and are available as options when creating a new document.

```
<section name="transactionSchemaSection" type="PTR.TransactionSchemaSection, PTR" />
 <Schemas>
 <Schema name="ExpereTxn" default ="true" displayName ="ExpereTxn" rootElement="Txn"
 targetNamespace="http://schemas.bankerssystems.com/2004/ExpereTxn" location="http://
schemas.bankerssystems.com/ExpereTxn/ExpereTxn.xsd"/>
 <Schema name="VMPMISMO" default ="false" displayName ="VMP MISMO" rootElement="LOAN"
 targetNamespace="http://schemas.bankerssystems.com/2005/VMPMISMO" location="http://
schemas.bankerssystems.com/Mortgage/VMP/VMP_Expere.xsd"/>
 <Schema name="QuickDoc" default ="false" displayName ="Quick Docs"
 rootElement="QuickDocs" targetNamespace="http://schemas.bankerssystems.com/QuickDoc"
 location="http://schemas.bankerssystems.com/QuickDocs/QuickDocs.xsd"/>
 <Schema name="custom_schema" default ="false" displayName ="CustomSchema"
 rootElement="CustomSchema" targetNamespace="http://schemas.bankerssystems.com/
custom_schema" location="http://schemas.bankerssystems.com/custom/custom_schema.xsd"/>
 </Schemas>
 </transactionSchemaSection>
```

> **Attention:**
> Adding a schema through the *Admin Tool* makes that schema available; it does not, however, imply the schema is valid. All schema files should be tested to ensure compatibility with *Expere* prior to being added to the solution.

Attribute	Description
name	The name of the schema, this must match the last part of the targetNamespace.
default	Define this as *true* to establish the schema as the default schema for the system.
displayName	The schema name as you want it to appear in the list of available options and within the application.
rootElement	The name of the root to use as a starting point for finding other elements.
targetNamespace	Specify the namespace that the schema targets, or validates.

Attribute	Description
location	The location, HTTP or file system, of the schema file (xsd).

Support for Predefined Packages

You can generate a document data requirement SCHEMA file when a predefined PKG file is present in the source repository. The SCHEMA file contains all the data points referenced by all the document package selection rules for each document referencing a given package. Packages correspond to phases defined in the phase group PHAS file.

Authoring Web Service Installation

Install and Configure the WebService Folder

1. Copy the **WebService** folder located in `C:\Program Files\WKFS\Expere Administration Tools\` to the Application Server.
2. Open the **WebService** folder on the Application Server.
3. Open **Web.Config** in Notepad.
4. Edit the server and database parameters in the connection string:

```
<connectionStrings>
<add name="RequirementsDB"
 connectionString="Server=<SERVERNAME>;Database=Requirements;
Trusted_Connection=True;Provider=SQLOLEDB;Integrated Security=SSPI;"
 providerName="System.Data.OleDb" />
```

```
<add name="SchemaRequestDB" connectionString="Server=
 <SERVERNAME>;Database=SchemaRequestDB;
Trusted_Connection=True;Provider=SQLOLEDB;Integrated Security=SSPI;"
 providerName="System.Data.OleDb" />
</connectionStrings>
```

Add Web Service to IIS

1. Open the Internet Information Services Management Console by selecting **Start>Programs>Administrative Tools>Internet Information Services**.
2. Click to expand the tree next to local computer and the Web sites folder.
3. Right-click **Default Web Site** and choose **New>Virtual Directory**.
4. Name the virtual directory **AuthoringWebService** and click **Next**.
5. Browse to the path of the **WebService** folder that was relocated to the Application Server.

6. Set or verify that the **Read and Run scripts (such as ASP)** permissions are allowed.
7. Click **Next**, accept the defaults until the wizard is complete.
8. Right-click the **AuthoringWebService** folder and select **Properties**.
9. Select the **Directory Security** tab and click **Edit…** under **Anonymous access and authentication control**. Verify that **Anonymous access** is checked and that the options in the **Authenticated access** area are unchecked. Click **OK** to close the *Authentication Methods* window.
10. Select the **ASP.NET** tab and set the **ASP.NET version** to **2.x** or greater.
11. Click **OK** to close the *Properties* window.
12. Verify the Authoring Web Service is working by opening Internet Explorer and entering the following URL: **http://localhost/AuthoringWebService/service.asmx**.

The Expere Administration Tool

The Expere Administration Tool (Admin Tool) allows you to:

- Set up an Expere Knowledge Base content repository containing Wolters Kluwer Financial Services' base content plus custom content developed in the Expere Authoring Environment's Requirements and Content Editor (Editor).
- Create and update the requirements database that is used by the Editor.
- Build content repositories.

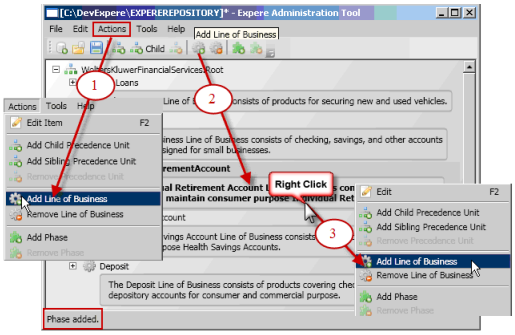
Administration Tool Basics

> **Note:**
> Windows 2008, R2 users should install SQL Server Express Management Console prior to using the Admin Tool.

The Admin Tool provides access to many standard functions using either the menus at the top of the application window, the right-click (options) menus, or toolbar buttons. For example, as shown below, you can add a line of business to the selected object from the Actions menu, the right-click menu, and using a toolbar button. The procedures in this user guide will generally use the right-click menu options. You can use whichever method is most comfortable for you.

When you select an object in the Admin Tool—for example, a Line of Business—it is highlighted and considered active. If a menu item or toolbar button is disabled, it cannot be used with the active object.

Status and confirmation messages for routine functions, such as adding and deleting objects, display in the lower-left corner (shown below) of the Admin Tool.



Users can access the **Add Line of Business** option from any of the following:

- **Actions** menu
- Toolbar
- Right-click menu

Expere Knowledge Base Repository

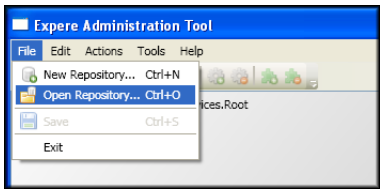
Overview

An Expere Knowledge Base repository contains content that, ultimately, is used to generate compliance documentation for financial transactions. For the purposes of this guide, there will be two repositories discussed: the authoring repository, where custom content is created and integrated with Wolters Kluwer Financial Services' content, and the build repository, which may be deployed to a production environment or to another environment such as testing. The content in the authoring repository will generally be under some type of source control or content management system that will place ("check out") files on the file system to be edited. The Admin Tool interacts with files in the file system.

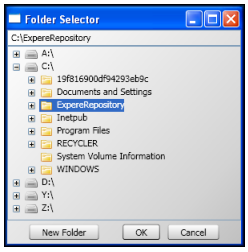
Opening an Existing Repository

With the Admin Tool running, open the Expere Knowledge Base repository by completing the following steps:

1. Select **File**/**Open Repository**.



2. The **Folder Selector** window appears.



3. Select the file path to the repository and click **OK**.

The repository you select will display in the Admin Tool.

You can configure load timeout to avoid failure loading huge repositories. The default timeout is 10 minutes.

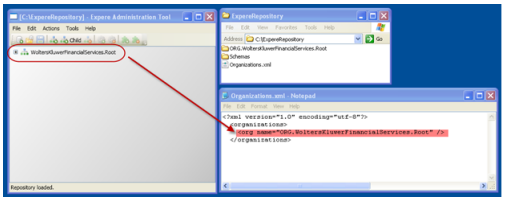
RequirementsAdmin.exe.config has key LoadTimeout in appSettings group:

```
<appSettings>
 <add key="LoadTimeout" value="10"/>
</appSettings>
```

If key "LoadTimeout" does not exist in config file, internal load timeout will be set to 10 by default.

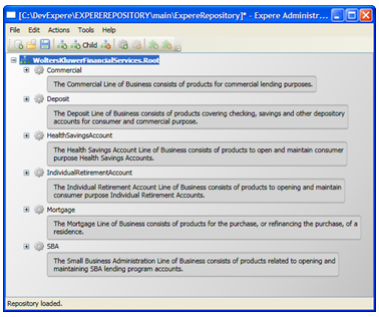
Expere Knowledge Repository Structure

When first installed, the standard Expere Knowledge Base repository structure consists of a file system folder containing a folder named WoltersKluwerFinancialServices.root, which contains Expere content organized in sub-folders, and an XML file, Organizations.xml, that contains information about the repository structure. With the Admin Tool open, select File > Open and select the folder containing the repository (the folder name is determined during the installation process and has no bearing on how the Admin Tool works). The authoring repository will open showing the WoltersKluwerFinancialServices.Root precedence unit. The Admin Tool is not displaying the file structure, but rather the information in the Organizations.xml file, as displayed below.

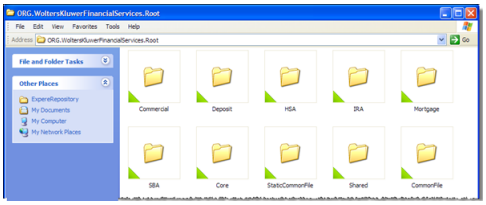


Secondary Structure: Lines of Business

Expere Knowledge Base content is organized by Lines of Business (LOBs). Each LOB is licensed independently; consequently, Wolters Kluwer Financial Services has organized the repository folder structure to contain a folder for each LOB. When WoltersKluwerFinancialServices.Root is expanded, the installed LOBs are displayed in the Admin Tool with the symbol.



The folder structure also contains “Common” and shared files that are used across LOBs.



Core Folders

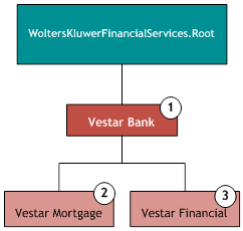
The document engine uses the core folder to render documents. It contains stylesheets and other files shared by all precendence units. You cannot override the core folder at custom precedence units as you can with regular .REQ content. Instead, add custom stylesheets to the core folder.

Custom Content and Precedence

Understanding Precedence

Precedence is a feature of the Expere Knowledge Base that allows custom content to supersede base content using a combination of inheritance and overrides. Precedence is applied by Precedence Units, which are discrete collections of content that are organized hierarchically. You may need to apply custom content to a specific Precedence Unit that will then be inherited by any child unit that does not override it.

Wolters Kluwer Financial Services is always the root (parent) Precedence Unit, at the top of the hierarchy and custom content resides in sub-levels below, as illustrated in the figure below. Any custom content in the Vestar Bank precedence unit will automatically apply to the Vestar Mortgage and Vestar Financial precedence units. Custom content in the Vestar Mortgage precedence unit will only apply to Vestar Mortgage and, likewise, custom content at Vestar Financial will only apply to Vestar Financial.



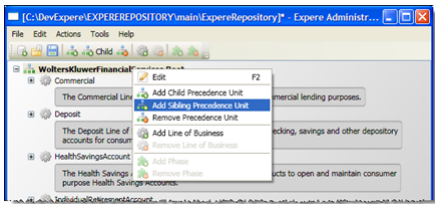
Working with Precedence Units

To create custom content, you will first need to create a precedence hierarchy in the Admin Tool. The Admin Tool edits the Organizations.xml file and constructs the file system folder structure where the content will be stored. When the repository is built, the custom content will be included in the build.

Add a Custom Precedence Unit

The following steps describe how to open a repository so you can add a new precedence unit or delete any obsolete units:

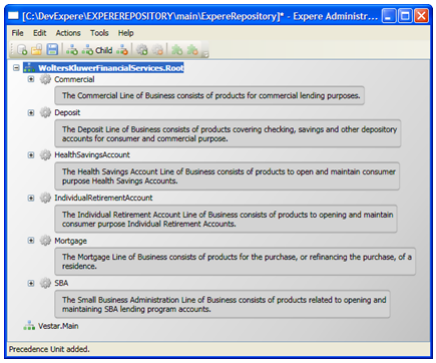
1. With **WoltersKluwerFinancialServices**.Root selected, right-click and select **Add Sibling Precedence Unit**.



   The new precedence unit is added and appears at the proper hierarchical level of the content structure.

   The precedence number is set by default and it increments each time a new unit is added, regardless of where the unit is added and whether the unit is a child or sibling.

2. Select the new precedence unit, right-click and select **Edit** to open an editing box (or press the **F2** key). Rename the unit, then click outside the box to save the name.



3. Add more precedence units as necessary.

4. 4. Select **File>Save**.

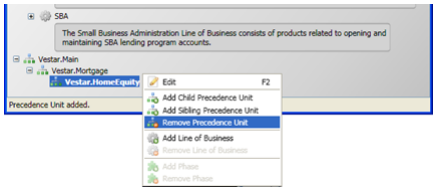
   In the Requirements Editor, you can create or save the custom content into the proper precedence folder in the repository folder structure on your system.

Remove a Precedence Unit

When you no longer need a precedence unit, it may be removed using the Admin Tool.

> **Note:**
> When removing a precedence unit, the folder, files within it, and all system links to that precedence unit and content are deleted.

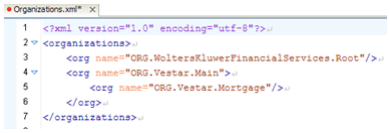
1. Select the Precedence unit to remove.
2. Right-click and select **Remove Precedence Unit**.



   The precedence unit will be removed from the Admin Tool precedence view.

3. Select **File>Save** to commit the changes.

   The Organizations.xml file will be updated and the precedence unit folder will be removed.



4. Select **File**>**Save** to save your changes.

Working With Lines of Business and Phases

To organize a specific group of transactions, or to categorize transactions with similar characteristics, the Admin Tool allows you to create a new Line of Business (LOB). For example, Home Equity is a LOB. The Home Equity LOB would consist of transactions, such as consumer lines of credit and the collateral used to secure the transaction, usually a personal residence.

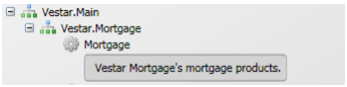
Add a Line of Business

In the Admin Tool, you can add lines of business and phases from the Actions menu, from the toolbar, or from the right-click menu. This procedure shows how to use the right-click menu options.

Use the following steps to add a LOB and phase:

1. Select the precedence unit level at which you want to add a LOB. Right-click and select **Add Line of Business**. A NewLineOfBusiness01 LOB will be added to the repository precedence view.
2. Rename the NewLineOfBusiness01 to a meaningful name.

   In this example, a Mortgage LOB is added to the Mortgage precedence unit.



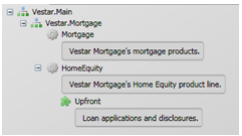
3. Select **File>Save** to commit the changes.

   When the LOB is added, the Admin Tool creates an XML PHAS (pronounced phase) file in the Precedence Unit's folder. The PHAS file is first added as PHAS.NewLineOfBusiness01.xml and its name is automatically updated when the LOB name is changed.

Add a Phase

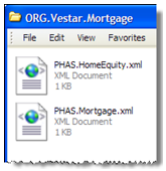
Within an LOB, you can create phases (logical groups) to break down the workflow by document type; each phase would contain a specific group of documents within a larger document set. Phases are used exclusively with LOBs; they cannot be added without a LOB. Unless an active LOB is selected, the menu and toolbar options for adding and removing phases are disabled. For the example below, another LOB, Home Equity, has been added as a child of Vestar Mortgage.

1. Select the LOB; right-click and select **Add Phase**.
2. Select the new Phase, press the **F2** key and change the phase name to **Upfront**.



3. Select **File>Save** to commit the changes.

   At this point, the Vestar.Mortgage folder will have two PHAS files: PHAS.HomeEquity.xml and PHAS.Mortgage.xml. The Home Equity phase file, PHAS.HomeEquity.xml, provides the LOB ID and description followed by the phase information.



Release Management

Topic paragraph

Introduction

The Admin Tool allows users to efficiently manage a series of content releases and updates over time. In particular, the release management features support users who implement WKFS standard updates and their own custom content, in addition to the various regulatory changes.

When updates occur, users can see what has changed at a WKFS base content level and at a custom content level. Users can make decisions about how changes get implemented. By being aware of the changes and their impact, users can implement content changes quickly and with confidence.

Release Management Best Practices

Effective release management requires that users follow the procedures outlined in the **Working With Build and Analysis Tools** section later in this guide. In summary, users always should generate the **Impact Analysis** report and, if merge conflicts exist, generate the **Interactive Merge** report to display and resolve the conflicts. In addition, users must read all relevant release notes and consult with subject matter experts to make the best decisions possible regarding content updates.

Getting Started

This section provides a conceptual overview of the Test Suite.

Opening Reports

To open any saved report or log file generated by the Admin Tool, click **File > Open Report**. Navigate to the file you want to display and click the **Open** button.

Working With Build and Analysis Tools

The Admin Tool includes utilities to perform the following:

- Building Expere content repositories in REQ and legacy ERL formats

- Generating Requirements and Content .xml*or databases
- Building and loading REQ and ERL formats to Expere Runtime
- Load to Expere Runtime

The Admin Tool also includes an **Impact Analysis Report**. When you generate the report it compares the new content to the existing content and will show the effects of updating that content. The Impact Analysis report summarizes the number of new and updated documents, and it identifies the automatic and any conflicted merge points. Conflicted merge points are highlighted, so you can find and click each one to see a description of the conflict (error).

In a.xml*ion to the Impact Analysis report, the Admin Tool includes the Interactive Merge Report. This report shows the content merge points and allows you compare the common base content to the existing and incoming content in one display.

Build the Requirements and Content Editor Database

After you make structural changes to a repository for custom content, for example: adding precedence units, you must build the database so the new precedence units can be displayed (and associated with documents) in the Editor.

> **Note:**
> If you do use custom content, you do not have to complete this procedure.

There is a command for building the content databases. You can run the command on-demand or set it up as a scheduled task. For more information, see the **Installation Guide for the Expere Authoring Environment**.

> **Note:**
> If you have made custom structural changes to a repository, you must generate the contents database for the Editor before creating any build.

To build a new content database, follow the steps below:

1. In the Admin Tool, open a repository by selecting **File**>**Open Repository**.

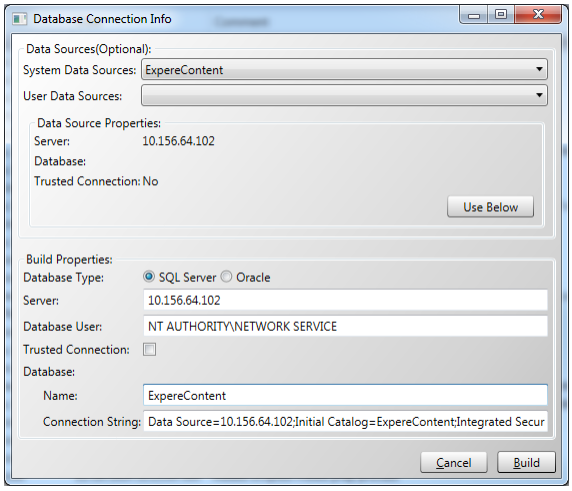
   The **Folder Selector** window appears.

2. Navigate to a repository and click **OK**.

   The repository opens in the Admin Tool.

3. Select **Tools**>**Generate Requirements & Content Editor Database**.

   The **DatabaseConnectionWindow** appears.



4. In the **System Data Sources** field, select **CAE Admin Tool**.

   The CAE Admin Tool data source creates during installation using the Windows Administrative Tools (ODBC). If this data source does not exist, create the data source using the Expere Authoring Environment installation procedures.

5. In the **User Data Sources** field, select **CAE Admin Tool**. Note that this was also created during installation.

6. Click the **Use Below** button to automatically complete the remaining fields on the window, based on the selected data source. The build saves to the server specified in the **Server** field (also specified during installation).

7. Enter **Requirements** in the **Name** field (under **Content Database**).

8. Click the **Build** button.

   The **Building Database** status bar appears in the lower-right corner of the Admin Tool. When the build is complete, a status message appears in the lower left corner.

9. Select **File**>**Exit** to close the Admin Tool.

After a successful database build, the changes made in the Admin Tool appear in the Editor. You can nowcreate a build of the repository.

Create a Content Build

After all the files for a content release have been updated in the Editor, and you have made the necessary structural changes to the repository in the Admin Tool, you can go ahead and create a build and release the content for testing, production, or another environment.

See **Expere Runtime Options** for more information about building content and loading it to different environments and platforms.

Also, there is a command to create a content build, as described below. You can run the command on-demand or set it up as a scheduled task. For complete details on how to use this command-line function and others, see **Using the Admin Tool Commands** in this user guide.

You can review and test assembled content in the Editor or in the Expere Testing System, which is a standalone testing tool for Expere content. See the Expere Testing System User Guide for more information.

> **Note:**
> You must generate the contents database for the Editor before creating any builds. See **Build the Requirements and Content Editor Database** for more information.

Procedure

Follow the steps below to create a content build:

1. In the Admin Tool, open the repository you want to release by selecting **File**>**New Repository**.

   The **Folder Selector** window appears.

2. Navigate to a repository and click **OK**.

   The repository opens in the Admin Tool.

3. Select a build option from the list of options in the Tools menu.

   The **Folder Selector** window appears.

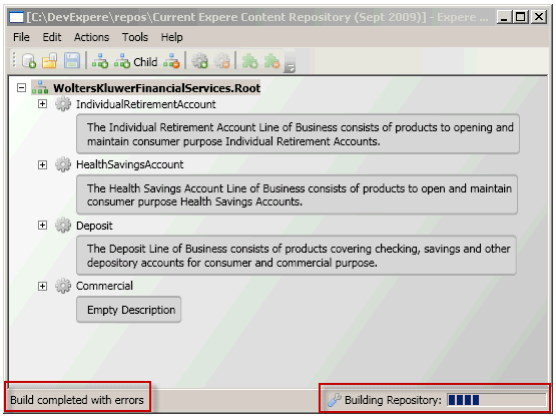
   > **Note:**
   > The remaining steps are identical for both build options.

4. From the **Folder Selector** window, select a folder for the build. Click the **New Folder** button to create a new folder for the repository (for example; **Builds**).

   If the file path you enter does not exist, a message prompts you to create the file path if desired.

5. Click **OK** to start the build.

   The Building Repository status bar appears in the lower-right corner of the Admin Tool. When the build is complete, a status message appears in the lower left corner.



6. The **Content Build** report appears when the build is complete. If any errors exist in the build, they appear in the report below **Failed documents**.
7. Scroll down in the report to view the failed documents. All error messages appear at the bottom of the report. Click **Failed** to display the error message for a document.
8. Using the .xml*or, fix the document errors listed in the report. Save each REQ file that you fix.
9. Return to the Admin Tool and repeat the steps above until the build finishes without errors.

Change the Content Build Default Time-Out

The time-out default for the content builder is 30 minutes. This value can be configured to a shorter or longer time in the Expere Admin Tool configuration file.

The time-out value is stored and can be changed in the root installation folder:

C:\Program Files\WKFS\Expere Administration Tools\ReqAdmin.exe.config

```
<?xml version="1.0"?>
<configuration>
 <startup>
 <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/></startup>
 <appSettings>
 <add key="BuildTimeout" value="30"/>
 </appSettings>
</configuration>
```

Using the Expere Runtime Load Option

The Admin Tool allows you to load Expere content to any environment. For example: Development, Integration Testing, Performance Testing, Quality Assurance, User Acceptance Testing, Training, Production, and so on.

The tool also allows you to load to popular database platforms, such as Oracle and SQL.

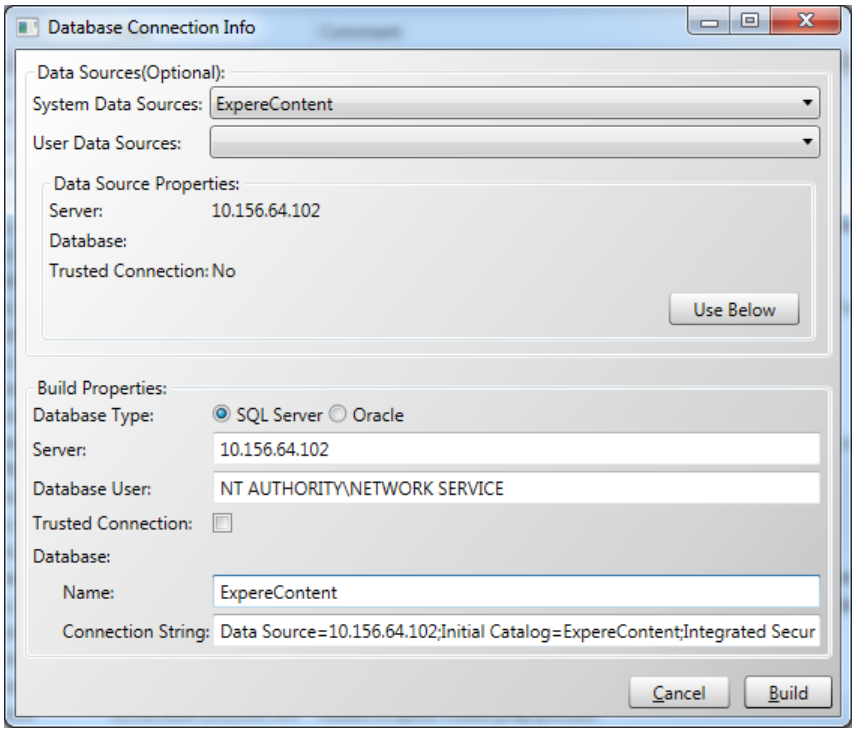
> **Note:**
> You must generate the contents database for the Editor before creating any build. See **Build the Requirements and Content Editor Database** for more information.

Procedure

Follow the steps below to Build and Load content:

1. Click the **Tools** menu to access the Expere runtime options.
2. Use the **Load to Expere Runtime** option to enter the existing target repository to load, and the target location for the finished build.
3. Complete the fields in the **Database Connection Info** window for the runtime option you selected.

   For example, enter an SQL connection string, target repository, and location for the finished build, as shown below.

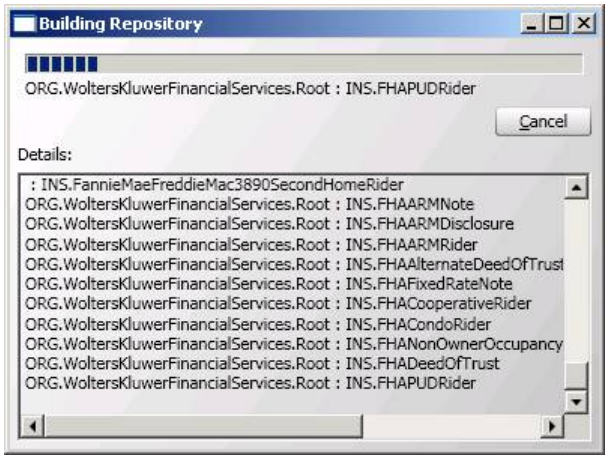


4. Click the **Build** button.

   The **Login** window appears.

5. Enter your server login credentials and click the **OK** button.

   The **Building Repository** window displays the build progress



6. When the build is complete, click the **OK** button to close the **Building Repository** window. You can now point to the build server location from any Expere instance.

Analyze and Load Content Updates

Before loading a new content repository, it is best to install the most current version of the Admin Tool, which is included on each content CD. Follow the instructions on the CD to first update the Admin Tool and then load the latest content.

Generate the Impact Analysis Report

When you want to analyze the specific differences between two content releases, open the existing content repository in the Admin Tool, select an incoming repository (usually the latest content release), run the Impact Analysis report.

The report includes the **Affected Documents** section, which details all the objects impacted by a change to legacy content and REQ items. Affected Documents appear for base and custom precedence levels, and the details include all documents that reference the changed REQ document or changed legacy content item. For example, you can see all the documents that use a common file that has been updated, which can help more fully understand the effect of the update.

Affected document reporting is for both base and custom documents and can be turned on or off for the Impact Analysis.

The Impact Analysis Report and Difference Decisioning and Merging window, discussed later in the procedure, show deleted content for both REQ and legacy objects, and they also display E-Signature fields, elements and attributes, if used.

> **Note:**
> The Impact Analysis report helps you to identify conflicts in the merged content, if needed. When necessary, seek the advice of a subject matter expert when you need to investigate the merge conflicts in depth.

Procedure

Follow the steps below to create an Impact Analysis Report:

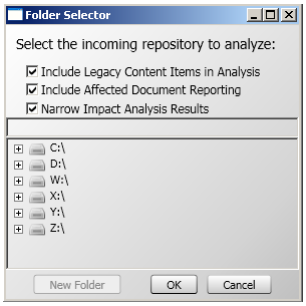
1. In the Admin Tool, open the existing content repository by selecting **File**>**Open Repository**.

   The **Folder Selector** window appears.

2. Navigate to the existing repository and click **OK**.

   The repository opens in the Admin Tool.

3. Select **Tools**>**Impact Analysis**. The **Folder Selector** window appears, allowing you to select the incoming (new) content repository.

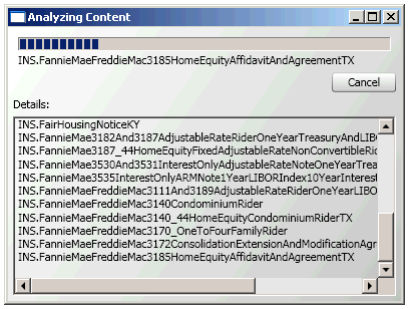


   The **Folder Selector** window allows you to select checkboxes for including Legacy Content Items and Affected Documents information in the report. Information for the option(s) you select displays in the Impact Analysis report, as well as in the Differences Tool, discussed later in this procedure.

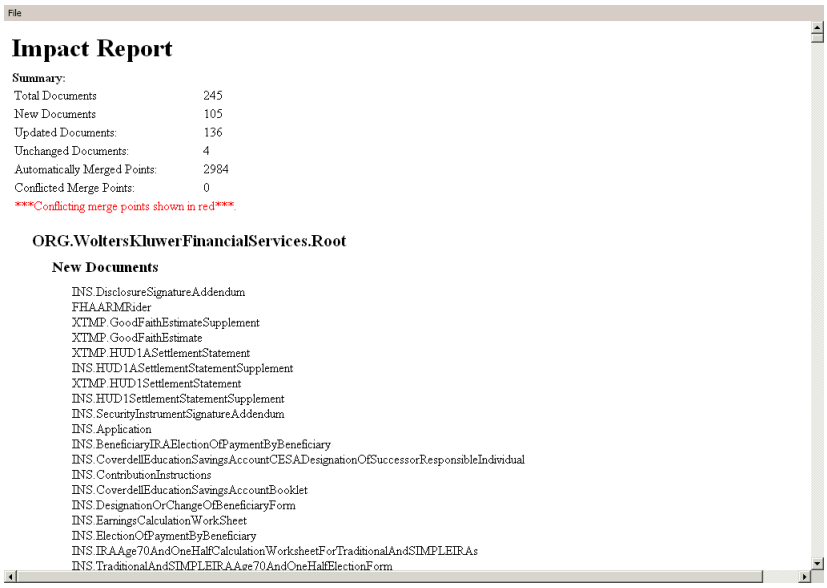
   If you select the **Narrow Impact Analysis Results** checkbox, the **Narrow Impact Results** window displays and allows you limit the Impact Analysis by selecting the precedence unit(s) and/or document(s) from the repository to analyze. On the **Narrow Impact Results** window, select the precedence unit(s) to include in the report.

4. Click **OK**.

   The **Analyzing Content** window appears. The status bar at the top indicates the report creation status. The **Details** box lists the files being analyzed.



When the analysis is complete, the **Build Complete** window displays the Impact Report.



5. If there are conflicting merge points, the report displays them in red. Scroll to find the conflicts and double-click each one to display the error message.
6. Resolve the merge errors and run the report again. Repeat until all errors are resolved.

Generate the Interactive Merge Report

When you want to examine all merge points, simultaneously comparing your content to existing base and incoming content, run the Interactive Merge report. You can step through each merge point one-by-one to analyze the impact of each content change, and then decide which version of the content to keep.

To ensure that the base precedence level for the incoming and existing repositories remain in synch, any content deleted from the incoming repository also is deleted from the existing repository when the merge process is completed.

Follow the steps below to run an Interactive Merge Report:

1. In the Admin Tool, open a content repository by selecting **File**>**Open Repository**.

The **Folder Selector** window appears.

2. Navigate to the existing repository and click **OK**.

   The repository opens in the Admin Tool.

3. From the **Tools** menu, select **Difference Decisioning & Merging**.

   The **Folder Selector** window appears, allowing you to select the incoming (new) content repository.

4. Click **OK** to begin the analysis.

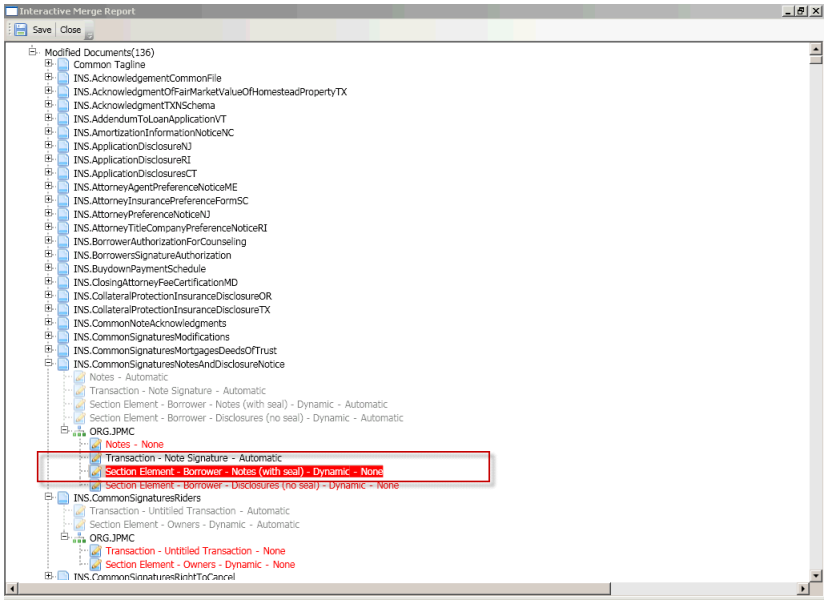
   The **Analyzing Content** window appears while the report is being generated.

5. The **Analysis Complete** dialog appears when the report is complete. This dialog indicates how many merge point conflicts were found.

6. Click **OK**.

   Regardless of the number of conflicting merge points, including 0, the Interactive Merge report (an HTML document) displays. Conflicts appear in red.

   You can use the relevant release notes for additional information and to help you troubleshoot merge conflicts.



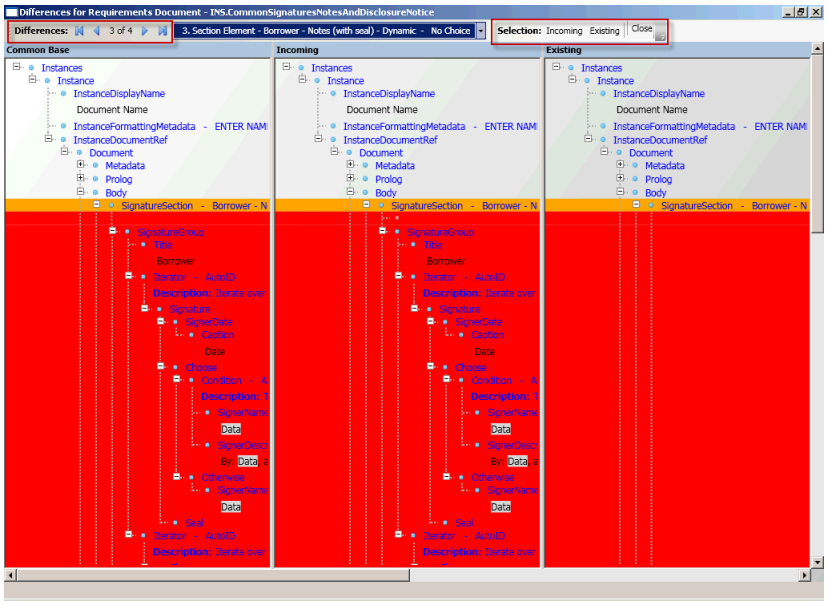
7. Click the **Save** button to save the report to your system for analysis.
8. Click the **Complete Merge** button to continue the merge process.

9. Click the **Create User Decisions Report** button to generate a report that includes all of the merge decisions. You cannot create this report until all merge conflicts have been resolved. This report is described in the Reports and Log section earlier in this guide.

10. Click the **Preferences** button to select a new display color for content differences and for highlighting all instances of the DTA, Iterator or Condition when using the **View References** button, described below.

11. Double-click a conflicted merge point to open the **Differences for Requirements Documents Tool (Differences Tool)**; this will allow you to analyze the merge points.

    If you double-click a COND, ITR or DTA, the Differences Tool will display the **View References** button. Use this button to display the References for: (DTA, condition or iterator) window, which highlights all instances of the DTA, Iterator or Condition in the REQ.



> **Note:**
> Be sure to click + to expand the comparison details; this allows you to see the detail of what has changed, including text changes and updated metadata, deletions and so on.

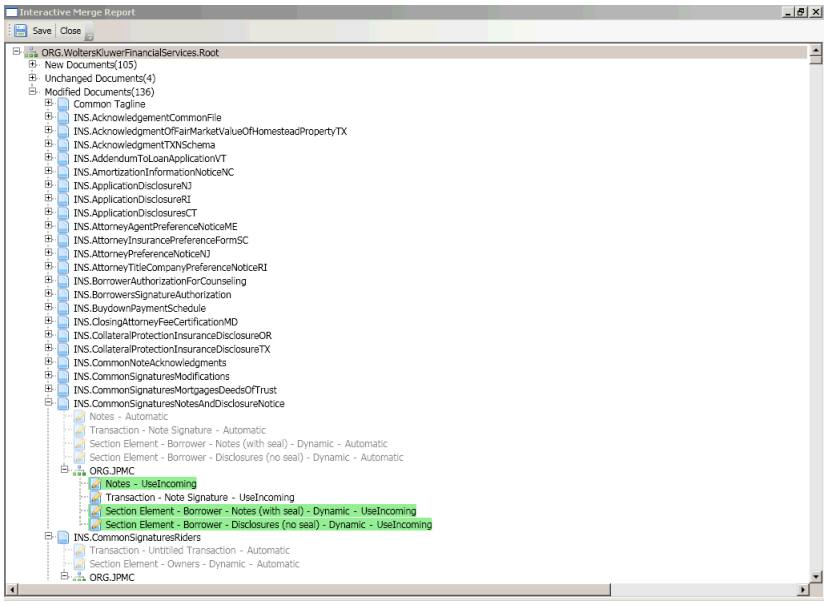
12. You can perform the following tasks from the Differences Tool:
    - Simultaneously display and compare the common base, existing, and incoming content
    - Select the content to use at a conflicting merge point by clicking either the **Existing** or **Incoming** button. These buttons exist in the **Selections** area of the tool. The window pane of the content you select is highlighted (in yellow) and is saved as you review other merge points.
    - Display the previous or the next merge point by clicking the back or forward buttons. Use these buttons to review the merge difference selections you have made before saving them. These buttons exist in the **Differences** section of the tool.
    - Save the merge point decisions by clicking the **Close** button (in the **Selections** area of the Tool).

- Display additional information and tips about content objects (in the red area of each pane): move your cursor over an object to display the information for that type of object.

> **Note:**
> If you need to stop before you complete the merge decision process, you can save your work and resume the process later. Save the file to your system. The file is given a *.merge extension. To resume the process, select **Tools > Continue Difference Decisioning & Merging** and select the saved file.

13. After you make your decisions and click the **Close** button, the Differences Tool closes.

    In the Interactive Merge report, the merge conflicts you have resolved now display in green.



14. Continue using the Interactive Merge report to identify the merge conflicts (displayed in red) so you can fix them using the Differences Tool.

15. After you have resolved every merge conflict listed in the report, click the Save button to save the report, or click the **Close** button to close without saving.

16. To ensure that all of the merge conflicts were addressed: select **Tools**>**Difference Decisioning & Merging**: run the report against the same repository until no merge conflicts exist.

Command Line Options

There are commands for performing Admin Tool tasks on-demand or on a scheduled basis. In addition there are unique commands for working with repositories and REQ files.

The majority of the command-line functions use a single executable, *ReqAdmn.exe*, that accepts the command-line parameters to determine which function to perform. In other cases a different executable

may be used; for example, generating dynamic document previews uses the BuildDocPreviews executable (*BuildDocPreviews.exe*).

The command-line functions described in this section are as follows:

- Building a repository
- Loading a repository (loading the output of a build to an Expere runtime environment)
- Building an authoring database
- Creating an impact report
- Preparing an REQ file for external delivery
- Trimming a repository to a single line of business
- Updating content references in dynamic documents
- Generating dynamic document outlines
- Generating dynamic document previews
- Database connection strings

Accessing the Command Prompt

To access the Windows command-line prompt (cmd.exe):

1. Perform either of the following:
   - Select **Start** > **Programs** > **Accessories**, and select **Command Prompt**.
   - From the **Start** menu, choose **Run**, enter **cmd**, and press **Enter**.

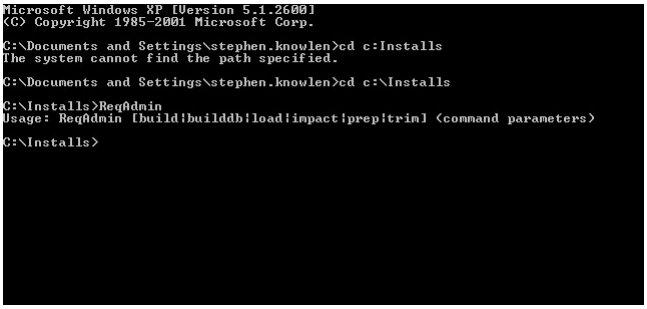
   The **Command Prompt** window will appear.

2. Once open, use the change directory (cd, or chdir) command to navigate to the **ReqAdmin.exe** folder location.

About Common Options

For a list of the Microsoft Window commands (similar to the MS-DOS commands) at the command-line prompt, type *help* and press **Enter**. A help list of the standard commands will appear.

When you invoke the command-line tool with no options, it displays a help message describing all of the command-line Admin Tool's usage functions. For example, type *c:\<your directory location> ReqAdmin* and press **Enter**.



As shown in **Building a Repository**, the command-line Admin Tool requires that you pass the first parameter as the name of the command to execute. The parameters that follow the first parameter will vary by the command entered.

For example, if you need details about the processing, enter the *-v (verbose)* option, and progress information will be written to *STDOUT*. Also, if errors:

- **Do not occur**, all commands implemented by the tool will not produce output to STDOUT.
- **Do** occur, they will be written to STDERR, and the tool exits with a non-zero exit code.

Building a Repository

The **build** command tells the tool to open the repository. It open the repository defined in the path to repository parameter. The command line to build a repository is as follows:

```
ReqAdmin build [-v] [-req] [-skipupdate] <Path to repository> <Output path>
```

The arguments, or switches, are optional. If the **verbose ([-v])** argument is entered, verbose logging output is used. If the **REQ [-req]** argument is passed, the REQ Expere Runtime loadable format is built from the source REQ files. If the REQ [-req] argument is not passed, the Expere Rule Language (ERL) format is built. If the **skip update [-skipupdate]** argument is passed, the repository is built without updating any content references. If the argument is not passed, content references in the target repository are updated before it is built.

The skip update switch is typically used when you have recently performed an update to content references using the **update** command just prior to building a content repository (using the **build** command). In this case, you are confident that the content references have been updated and are not out of date. See *Updating Content References in Dynamic Documents* for details.

> **Note:**
> The **build** command will update content references from the content in the file system repository that is being built; this differs from behavior in the Requirements Editor where content reference updates are performed through the authoring web service and database.

The output is then placed at the path location entered in the second path parameter: <*Output path*>. The directory output layout, as required by the load command, is as follows:

```
<output root>
 - Organization 1 name
 -REQ or ERL files
 - Organization 2 name
 -REQ or ERL files
 - Organizations.xml
 - BuildReport.html
 - BuildReport.xml
 - DocumentRevisionHistory.csv
```

Both the *BuildReport.html* and *BuildReport.xml* files contain the results of the build and detail any REQ files that failed to build. These two files contain identical information, only in different formats: one in xml and one in html (recommended for viewing). The *DocumentRevisionHistory.csv* file contains the cumulative revision history of all the REQ files that were built.

Loading a Repository

The **load** command takes the output layout from the build and loads it to an Expere runtime environment. This process connects directly to the database, so you must enter a database connect string at the end of the command. See *Database Connection Strings* for details.

The command line to load a repository to Expere is as follows:

```
ReqAdmin load [-v] (-sql|-oracle) <Path to build output> <Database connection string>
```

The type of database, where the content was loaded, must be entered on the command line by typing either an *–sql* or *–oracle* switch. If you do not enter one of these switches, the tool reports an error to *STDERR* and terminates with a non-zero exit code.

The load command causes the Expere runtime environment to mirror the repository indicated. It also synchronizes the organization structure in the runtime with the local repository, and matches all of the content to what is local.

The load command function may be either run against a runtime environment that already has content, or a fresh install. In either case, after running the load command, all of the content that was in the repository will now be accessible by the document engine.

You can configure load timeout to avoid failure loading huge repositories. The default timeout is 10 minutes.

ReqAdmin.exe.config has key LoadTimeout in appSettings group:

```
<appSettings>
 <add key="LoadTimeout" value="10"/>
```

```
</appSettings>
```

If key "LoadTimeout" does not exist in config file, internal load timeout will be set to 10 by default.

Building an Authoring Database

The **builddb** command function rebuilds an authoring database instance to match the repository indicated. The command line to build an authoring database that mirrors a repository is as follows:

```
ReqAdmin builddb [-v] <Path to repository> <Requirements Database connection string>
 <Schema Request Database connection string>
```

This command connects directly to the target database, and that means you must enter a connect string. For details about connect strings see *Database Connection Strings*.

Once the command is run, the authoring web services reports the updated information (that matches the content in the local repository) to that which the command was run against.

Creating an Impact Report

The **impact** command takes both the new repository and the old repository, then generates a report that details the effect of merging the new repository into the old repository. The command line to build an impact report is as follows:

```
ReqAdmin impact [-v] [-xml] <Path to incoming (new) repository> <Path to existing
 (old) repository> [Path and name for output file]
```

By default, the impact command function produces an HTML file with all of the results; however, an XML file may be produced instead by entering the *-xml* option.

Additionally, the report writes to an *impact.[html/xml]* file, but you can override this by giving a path and file name following the path to the old repository.

Data Dictionary Builder

The Data Dictionary Builder is the specific tool used to create a new data dictionary.

The command line to build a new data dictionary is as follows:

```
DataDictionaryBuilder {-create | -validate | -continue [-ddOutputPath path]}
[-script filename]
[-copyRight copyRigtString]
[-url endPointUrl]
[-buildFromEndpoint]
[-orgAdd o/rgName name /orgID ID /parentOrgName parentOrg /orgDescription description]
[-orgContent /OrgName name /path ContentPath]
[-orgRequirements /makeOrgFolders /inlcudeChangeNotes /OrgName name /path
 contentPath /outlinePath outlinePath]
[-xldd path]
[-attributes schemaPath]
[-addXpaths]
[-enumDesc lddPath]
```

```
[-sampleData path]
[-addOPPSA]
[-addDocViewerAttribs]
[-validateOrgStructure]
[-validateAttributes]
[-validateRequirements]
[-validateAutoselection]
```

- **-create** Creates new data dictionary
- **-validate** Turns on all validate switches
- **-continue** Opens the data dictionary for continued building
- **-ddOutputPath** Specify to save to a different location
- **-ddPath** Path of data dictionary
- **-script** Specify a script with switches
- **-copyRight** Copyright string
- **-url** End Point Url
- **-buildFromEndpoint** Builds by pulling content from an end point
- **-orgAdd** Adds an org to the precedence structure
- **-orgContent** Builds by pulling content from file system
- **-orgRequirements** Creates DocumentRequirements tab imports REQ files
- **/makeOrgFolders** Specifies to separate PDF outlines by org folder, to preserve different versions of by org
- **/inlcudeChangeNotes** Specifies to inlcude change notes when importing REQ files
- **/OrgName** Specifies the org to import REQ files into
- **/path** Specifies the path to find REQ files to import
- **/outlinePath** Specifies the path to find PDF outlines to import
- **-xldd** path to the MS Access db for additional attribute metadata
- **-attributes** Imports additional attributes from xldd
- **-addXpaths** Imports remaining schema
- **-enumDesc** Imports enumeration descriptions from the LDD
- **-sampleData** Imports sample attribute values
- **-addOPPSA** Imports OPPSAFlag attribute
- **-addDocViewerAttribs** Imports SuppressUIFlag, OrderIndex and GroupName attributes
- **-validateOrgStructure** Validate each orgs packages and documents against an Expere repository URL
- **-validateAttributes** Validates attributes against the schema

- **-validateRequirements** Validates each document in the dictionary has requirements
- **-validateAutoselection** Validates autoselection rules against an Expere repository URL.

Preparing an REQ File for External Delivery

The **prep** command takes in a single REQ file, a semi-colon separated list of files, or a directory, then prepares the REQ files found for external delivery. The command line to prepare a REQ file for external delivery is as follows:

```
ReqAdmin prep [-v] (<Path to file>[;<Path to file>]* | <Directory>)
```

When a directory lists, it is assumed that all subdirectories are included and all **.req* files in them are subject to prepping as well. Preparing for delivery entails the following:

- notes from the notes tab are removed
- profile instances are removed
- saved transactions are removed
- content that is marked for output format, other than runtime, is removed

These changes are made to the REQ files in-place; so, the result of running this command is that all of the files listed are modified, as listed above.

Trimming a Repository to a Single Line of Business

The **trim** command can be used against a repository on the file system to eliminate all of the content unrelated to a particular line of business. The command to trim a repository to a single line of business is:

```
ReqAdmin trim [-v] <Repository path> <Line of business id>
```

Running this command opens a repository, inspects every content item and REQ file, and (if it was not used by the line of business specified) it will be removed.

Updating Content References in Dynamic Documents

The **Update** command can be used, for example, after the Diff/Merge process and before generating document previews to ensure all content references are up to date. The command to update all dynamic REQ content references within a repository is as follows:

```
ReqAdmin update [-v] <Repository Path>
```

> **Note:**
> The command will update content references from the content in the file system repository that is being built; this differs from behavior in the Requirements Editor where content reference updates are performed through the authoring web service and database.

A typical workflow might involve performing an update to content references using this command (**update**) prior to building a content repository (using the **build** command). In this workflow, you may elect to use the *-skipupdate* switch in the **build** command to bypass the update process that is default behavior as part of the build repository. See *Building a Repository* for details.

Generating Dynamic Document Outlines

The command to generate dynamic document (static document outlines not currently supported) outlines for an entire Expere repository lets you create outlines based on REQ files in a given source repository. This is a convenient method to generate outlines without having to open each REQ file individually in the Requirements Editor.

The syntax for the command looks like:

```
<REQ file directory> -output <output directory> -log <path to log file> -errorLog
 <path to error log file>
```

- <REQ file directory> is the file path of the Expere repository.
- <output directory> is the desired output file path for the document outlines.
- <path to log file> is the desired path for the outline build log file.
- <path to error log file> is the desired path for the outline build error log file.

> **Note:**
> Be sure to update all content references in the repository to ensure the outlines are current.

About the CustomerFacingOutline Application
You must install the CustomerFacingOutline executable to run the command line tool. You can find the *CustomerFacingOutlineInstaller/setup.exe* and all related files in the following directory:

```
<Release Number>_Release/Tools/CustomerFacingOutline
```

Examples

- Build Outlines:

  ```
  C:\Temp\CustomerFacingOutline>CustomerFacingOutline C:\ExpereRepository
  \ORG.WoltersKluwerFinancialServices.Root\Mortgage –output C:\mortgageoutlines
  ```

- Specified Expere REQ File Directory:

  ```
  C:\ExpereRepository\ORG.WoltersKluwerFinancialServices.Root\Mortgage
  ```

- Specified outline output path:

```
C:\MortgageOutlines
```

Generating Dynamic Document Previews

The **BuildDocPreviews** command to generate dynamic document (static document previews not supported) previews for an entire Expere repository lets you create a preview (PDF or RTF output) based on REQ files in a given source repository. This is a convenient method to generate previews of output PDFs without having to open each REQ file individually in the Requirements Editor.

Using the command line, dynamic document previews are generated through batch processing and are placed in the directory specified by the user. Each preview is named according to its document name. Document previews include most up-to-date content references (assuming content reference files were updated prior to processing).

The command line to generate dynamic document previews looks like:

```
BuildDocPreviews (-FOP|-TAG|-XEP|-RTF) <Repository Path> <Output Path>
```

The arguments, or switches, available include the following. These switches are optional and mutually exclusive:

- -TAG generates tagged PDF previews with Apache FOP. Use the TAG switch to generate the document(s) as an ADA-enabled TaggedPDF using the Apache FOP rendering engine.
- -FOP generates straight PDF previews with Apache FOP.
- -XEP generates straight PDF previews with RenderX
- -RTF generates RTF previews. Use the RTF switch to generate the document as an RTF that can be opened in Microsoft Word.

> **Note:**
> If an assembly mode is not specified, the default is FOP.

The target repository folder and output folder are defined in the following parameters.

- <Repository Path> is the file path of the Expere content repository containing the dynamic REQ format documents. This must be a fully qualified path.
- <Output Path> is the desired output file path where the document previews are written to. This must be a fully qualified path. In addition, a *BuildPreviewsLog.txt* file that lists the successfully generated documents, any static document names (previews of static documents are not supported), as well as document rendering errors are written to this folder.

> **Note:**
> The best practice recommendation is to avoid using spaces in the repository or output folders. If either of these folders contain spaces, you must enclose the folder path with quotes (" ") in the command line.

You must CD to the BuildDocPreviews install folder and run the command line prompt from the install folder. In the source repository, make sure the organization structure file (organizations.xml) is contained

at the root of the repository folder. As well, be sure to update all content references in the repository to ensure the previews are current.

> **Important:**
> You must install the BuildDocPreviews application on the target machine in order to use the command line to generate a document preview.

About the BuildDocPreviews Application
As noted, you must install the BuildDocPreview executable to run the command line tool. You can find the *BuildDocPreviewInstaller/setup.exe* and all related files in the following directory:

```
<Release Number>_Release/Tools/BuildDocPreviews
```

> **Note:**
> The Expere Authoring Tools CD distributed for use with the Apache FOP rendering engine does not support the -XEP switch or Render X as the document rendering option.

If you intend to generate previews using the Render X rendering engine, by using the -XEP switch, then you must install the BuildDocPreviews installation program distributed on the XEP (RenderX) version of the Expere Authoring Tools CD.

Example

In this example, the BuildDocPreviews application was installed to the default installation folder and the repository and output folder are at the c:\ root.

The following generates previews based on the repository using the FOP rendering engine.

```
C:\Program Files (x86)\WKFS\BuildDocPreviews>BuildDocPreviews -FOP c:\ExpereRepository
 c:\ExpereRepositoryPreviewsOutput
```

Database Connection Strings

A database connection string provides an application with all of the information needed to connect to a database.

This information depends on the type of database, but usually includes a user name, password, server name, database name, and so on.

For the SQL Server, a typical connection string may resemble the following:

```
"user id=sa;password=sql;server=localhost;database=expere;"
```

For Oracle, a typical connection string may resemble the following:

```
"Data Source=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=mn02ws1finnelb)(
PORT=1521)))(CONNECT_DATA=(SERVER=DEDICATED)(SID=orcl)));User
Id=fw3;Password=fw3;"
```

The keys in the connection strings (user id, password, server, data source, etc.) are not case sensitive.

For details on the options available, please refer to the documentation from your database vendor.

Removing Orphaned/Unused Rules

The **removeorphanedrules** command is a function that removes unused rules from REQs in the specified repository. Prior to the creation of this function, the only way to remove unused rules is by opening a single REQ file in the RE Tool and executing the Build function, which then prompts you to remove unused rules. This function can be used to remove unused rules for all REQ files in a given repository. The command line to build a repository is as follows:
ReqAdmin removeorphanedrules <Path to repository>

About Wolters Kluwer Financial Services - Whether complying with regulatory requirements or managing financial transactions, addressing a single key risk, or working toward a holistic enterprise risk management strategy, Wolters Kluwer Financial Services works with more than 15,000 customers worldwide to help them successfully navigate regulatory complexity, optimize risk and financial performance, and manage data to support critical decisions. Wolters Kluwer Financial Services provides risk management, compliance, finance and audit solutions that help financial organizations improve efficiency and effectiveness across their enterprise. With more than 30 offices in 20 countries, the company's prominent brands include: FRSGlobal, FinArch, *ARC* Logics®, TeamMate®, Bankers Systems, VMP® Mortgage Solutions, AppOne®, GainsKeeper®, Capital Changes, NILS®, AuthenticWeb™ and Uniform Forms™. Wolters Kluwer Financial Services is part of Wolters Kluwer, a leading global information services and solutions provider with annual revenues of (2012) €3.6 billion ($4.6 billion) and approximately 19,000 employees worldwide. Please visit our website for more information.

Wolters Kluwer Financial Services
6815 Saukview Drive
St Cloud, MN, 56303
Toll-free: 800.274.2711

To learn more visit **WoltersKluwerFS.com.**

Wolters Kluwer
Financial Services