



Managing Repositories

For Java

FICO® Blaze Advisor® decision rules management system

Version 7.3

This document is the confidential, unpublished property of Fair Isaac Corporation. Receipt or possession of it does not convey rights to divulge, reproduce, use, or allow others to use it without the specific written authorization of Fair Isaac Corporation and use must conform strictly to the license agreement.

The information in this document is subject to change without notice. If you find any problems in this documentation, please report them to us in writing. Fair Isaac Corporation does not warrant that this documentation is error-free, nor are there any other warranties with respect to the documentation except as may be provided in the license agreement.

© 2000–2015 Fair Isaac Corporation. All rights reserved.

Permission to use this software and its documentation is governed by the software license agreement between the licensee and Fair Isaac Corporation. Portions of the program may contain copyright of various authors and may be licensed under certain third-party licenses identified in the software itself in the <ADVISOR_HOME>/doc/thirdPartyLicenses/Readme.pdf file.

In no event shall Fair Isaac be liable to any person or direct, indirect, special, incidental, or consequential damages, including lost profits, arising out of the use of this software and its documentation, even if Fair Isaac has been advised of the possibility of such damage. The rights and allocation of risk between the licensee and Fair Isaac are governed by the respective identified licenses in the <ADVISOR_HOME>/doc/thirdPartyLicenses/Readme.pdf file.

Fair Isaac specifically disclaims any warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The software and accompanying documentation, if any, provided hereunder is provided solely to users licensed under the Fair Isaac Software License Agreement. Fair Isaac has no obligation to provide maintenance, support, updates, enhancements, or modifications except to users licensed under the Fair Isaac Software License Agreement.

FICO™, Fair Isaac and Blaze Advisor® are trademarks of Fair Isaac Corporation in the United States and may be trademarks or registered trademarks of Fair Isaac Corporation in other countries. Other product and company names herein may be trademarks of their respective owners.

FICO™ Blaze Advisor® decision rules management system is covered by U.S. Patents Nos. RE43,474; 6,865,566; 6,965,889; 6,968,328; 6,933,514; 7,000,199; 7,152,053; 7,277,875; 7,428,519; 7,613,671; 7,831,526; 7,835,932; 7,930,196; 7,937,355; 8,099,376; 8,200,609; 8,237,716; 8,280,836; 8,730,241 and Indian Patent No. 240176 and others listed on www.fico.com.

FICO® Blaze Advisor® 7.3

Last Revised July 31, 2015

Version 7.3

Template LG 6.2.3

Contents

CHAPTER 1

Designing a Blaze Advisor Repository 13

Blaze Advisor Repository Overview	13
Key Decision Points for Choosing a Blaze Advisor Repository	14
Characteristics of Repository Types and Suggested Usage.....	15
Deciding on a Database Type for your Repository	16
Source Control Management.....	17
Security Considerations.....	19
Blaze Advisor Authentication	19
Blaze Advisor Authorization	20
Blaze Advisor Content At Rest Management	21
The Blaze Advisor Repositories and Workspaces	21
The Repository Structure	22
Storage Names and Names in the Blaze Advisor IDE	23
Repository Entities.....	24
Repository Administration.....	24
Implementing Repositories	25

CHAPTER 2

Implementing a Blaze Advisor Repository..... 27

About Creating a Blaze Advisor Workspace or Repository	27
Creating a Blaze Advisor Repository	28
Creating a Repository Using the Blaze Advisor IDE	28
Creating a Blaze Advisor Repository Using the Utility.....	33
Writing the Connection Configuration File for an Admin Repository	33
Writing the Connection Configuration File for a File Repository	34
Writing the Repository Configuration File for a File Repository	34
Writing a Batch File to Create the Repository Using the Utility	35
Removing a Blaze Advisor Repository Using the Utility	37
Writing a Batch File for Removing a File Repository	37
Blaze Advisor Repository Contents and File Structure	37
Editing a Repository Configuration in the Blaze Advisor IDE	39
Saving User Name and Passwords	43
Password Encryption with the Encryption Manager	44
Adding Encryption Manager to an Existing Repository	46
Using NdRomAdminUtil to Encrypt a Password	48
Allowing Distinct Repository and Workspace Users	49
Converting a Repository to Use Another Type	50

CHAPTER 3**Connecting to a Blaze Advisor Repository 51**

Workspace and Repository Connections	51
Connecting to a Blaze Advisor Workspace or Repository	52
Connecting to an Existing Repository.....	52
Selecting a Repository or Workspace Connection	53
Connecting to a Blaze Advisor Repository From a Rule Maintenance Application	55
Connecting to a Blaze Advisor Repository From a Deployed Rule Service.....	56
Network Connection and Repository Type Considerations For Rule Service Deployment Performance	57
Creating, Modifying or Deleting Blaze Advisor Repository Connections	57
Creating a New Blaze Advisor Workspace or Repository Connection.....	58
Determining the Workspace Type for Your Connection	59
Modifying a Blaze Advisor Workspace or Repository Connection	59
Deleting a Blaze Advisor Repository or Workspace Connection.....	60
Importing a Blaze Advisor Workspace or Repository Connection	61
Exporting a Blaze Advisor Workspace or Repository Connection	62
Editing a Generated Connection File.....	63

CHAPTER 4**Administering Blaze Advisor Workspaces and Repositories .. 65**

Importing or Exporting Projects and Workspace Contents Using the Utility	65
Importing a Project from an .adv File Using the ImporterExporter Utility.....	66
Connection Parameter Sub-arguments	67
Importing a Project from a Zip File Using the Utility	68
Exporting a Project as an .adv File Using the ImporterExporter Utility.....	69
Connection Parameter Sub-arguments	70
Exporting a Project to a Zip File Using the Utility	71
Exporting and Importing a Project.....	72
Importing a Workspace from a Zip File Using the Utility	74
Exporting a Workspace to a Zip File Using the Utility	74
Exporting and Importing a Workspace Using the Utility	75
Releasing a Project Using the Utility	76
Setting a Release Manager.....	78
Showing a Release Manager.....	78
Publishing a Project Using the Utility	79
Updating a Published Project Using the Utility	81
Replacing a Published Project Using the Utility.....	82
Managing Versioned Project Content Using the Utility	83
Checking In New or Existing Items Using the Utility	84
Checking Out One or More Items Using the Utility	85
Canceling the Check Out for One or More Items Using the Utility	86
Deleting a Versioned Item Using the Utility	87
Restoring Logically Deleted Items	88
Viewing Item History Using the Utility.....	89
Checking the Status of Workspace Files Using the Utility	90
Promoting an Item Using the Utility	91

Updating a Private or Local Workspace Using the Utility.....	91
Adding Labels to a Workspace Entry.....	92
Retrieving Labels for a Workspace Entry.....	93
Deleting a Label from a Workspace Entry	94
Managing Files Using the Utility	95
Obtaining a List of Locked Files	95
Unlocking Files by User.....	96
Adding a Query	97
Removing a Query	98
Listing Queries	99
Applying Fixes.....	99
Adding Management Properties to a Repository	101
Management Properties.....	101
Adding a Management Property to an Entity Category.....	102
Viewing a Management Property in an Editor	103
Editing a Management Property	104
Removing a Management Property	105
Predefined Management Properties	105
Entity Categories	109
Creating a Management Property	110
Setting the Advanced Display Options for a Management Property	113
Advanced Display Options	114
Creating a Management Property Value List	116
Displaying Management Properties in a Folder or Project Editor.....	117
Removing Management Properties from a Project or Folder Editor	118
Cutting, Copying, and Pasting Project Items with Management Properties	118
Managing Templates and Instances in Your system Folder	119
Displaying the system Folder.....	119
system Folder Contents	120
Management Properties Library	120
Query Library	121
Management Property Templates	123
About Management Property Templates	123
Creating a New Management Property Template	124
Editing a Management Property Template	126
Deleting a Management Property Template	126
Adding a Management Property to a Category in the system Folder	127
Management Property Types.....	128
About the Management Property Types	128
Management Property Providers	129
Schema Management of Type Information	130
Managing Repository Configuration and Connection Instances in the Admin Repository	131
Admin Repository	132
Connecting to the Local Admin Repository	132
Contents of the Local Admin Repository.....	133
Configuration Templates	133
Configuration Instances	133
Connection Templates	133

Connection Instances	134
Editing Configurations and Connections Instances.....	134
Editing Configuration Instance Files	134
Editing Connection Templates.....	135
Changing the Password Display in an Existing Connection Template	135
Editing the Directory Path in an Existing Connection Template	135
Editing Connection Instance Files.....	136
Creating a New Connection Instance to use a Database Private Workspace	136
Manually Creating a Connection Based on an Existing Connection Instance	136
Deleting Connection Instance Files	137
Synchronizing the system Folder	137
How Modifications to Templates and Instances in the system Folder are Treated by the Synchronization Process	137
Prerequisites for Synchronizing the system Folder of a Non-versioned Repository	138
Prerequisites for Synchronizing the system Folder of a Versioned Repository	138
Running the Utility to Synchronize a system Folder	139
Updating Workspaces Connected to a Synchronized Repository	142
Changing an Existing Repository Configuration	144
About the ROM Configuration File Provided with Your Blaze Advisor Installation	145
Changing the Configuration of a Non-versioned Repository.....	146
Changing the Configuration of a Versioned Repository.....	148
Committing a Configuration Change from a Workspace to a Repository.....	150
Updating a Configuration Change from a Repository to a Workspace	151
Loading Custom Classes	152
Creating Self-contained Configurations.....	155
Embedding a Digital Signature Into a Configuration	157
Verifying a Digital Signature	159

CHAPTER 5

Repository Administration Reference 161

The NdRomAdminUtil Utility	161
The NdRomAdminUtil Utility Commands	162
Obtaining Help from the Command Prompt.....	165
Running the NdRomAdminUtil Utility	166
Entering a Symbolic Name or a Pathname as an Argument Value	166
Entering a Pathname	167
Entering a Symbolic Name	167
About Writing Files for Creating a Blaze Advisor Repository or Workspace Using the Utility	168
Writing Connection Configuration Files	169
Sample Connection and Configuration Files Installed With Blaze Advisor.....	170
Repository Connection Configuration File Examples	171
File Repository	172
Database Repository	173
Repository with BVS Versioning	175
File Repository with CVS Versioning	177
File Repository (Subversion versioning)	179
File Repository (ClearCase versioning)	181
Saving User Names and Passwords Using the Utility	183

Allowing Distinct Repository and Workspace Users When Using the Utility	184
Writing a Connection Configuration File for a Private or a Local Workspace.....	185
File Workspace	186
Database Workspace	187
Writing a Repository Configuration File	191
Adding a Version Manager	193
Adding an Authorization Manager.....	195
Adding an Authentication Service for IP Protection	196
Adding a Repository Entry Filter	197
Adding an Item Content Converter.....	198
Adding a Query	199
Adding a Filter	200
Example Configuration Files	201
Repository Configuration With No Services	202
Repository Configuration with BVS Versioning	202
Repository Configuration with File CVS Versioning	204
File Repository (Subversion Versioning)	205
File Repository (ClearCaseVersioning)	206

CHAPTER 6**Implementing a Repository With a Blaze Versioning Service. 209**

Deciding on the Type of BVS Workspace	209
Private Workspaces.....	211
File Structure for a Private or Local Workspace	212
Shared Workspaces.....	213
File Structure for a BVS Shared Workspace	214
Creating a BVS Repository Using the Blaze Advisor IDE	216
Creating a File Repository With BVS Private Workspaces Using the Blaze Advisor IDE.....	217
Creating a File Repository With a BVS Shared Workspace Using the Blaze Advisor IDE.....	219
Creating a File BVS Repository Using the Utility	220
Creating a File BVS Repository With Private Workspaces Using the Utility	221
Writing a Connection Configuration File for a File BVS Repository	221
Writing a Repository Configuration File for a File BVS Repository	223
Writing a Batch File to Create a File BVS Repository	224
Creating a File BVS Repository With a Shared Workspace Using the Utility	226
Creating a Local or Private Workspace	226
Creating a Workspace in the Blaze Advisor IDE	227
Creating a Blaze Advisor Workspace Using the Utility.....	228
Writing a Connection Configuration File for a Workspace	228
Writing a Batch File to Create a Workspace	229
Removing a Workspace Using the Utility.....	229
Writing a Batch File to Remove a Workspace	230
Creating BVS Workspace Connections	230
Verifying Blaze Advisor Versioning Commands	230
Converting a Blaze Advisor Repository to Using a Blaze Versioning (BVS)	232

CHAPTER 7**Implementing a CVS Repository 235**

Setting Up a CVS Server	236
Verifying Your CVS Server Installation	236
Creating a Repository Directory on a CVS Server	236
Creating a Repository Directory	236
Creating a Module in the Repository Directory	237
Creating a File CVS Repository Branch	237
Using a Password-authenticating Server	237
Using Access Permissions Supported by Blaze Advisor	238
Local Workspaces	238
Creating a CVS Repository	239
Creating a CVS Repository in Blaze Advisor	239
Creating a CVS Repository Using the Utility	242
Writing the Connection Configuration File for the File CVS Repository	242
Writing the Repository Configuration File for the File CVS Repository	244
Writing a Batch File for Creating the File CVS Repository	245
Creating Workspaces and Workspace Connections for a CVS Repository.....	246
Converting a Repository to and from a File CVS Repository	247
Creating a CVSNT Repository	248
Prerequisites for Creating a CVSNT Repository	248
Writing a Connection Configuration File for a CVSNT Repository	248
Writing a Connection Configuration File for the system Folder	249
Editing the Repository Configuration File	249
Running a Batch File to Create a CVSNT Repository	250

CHAPTER 8**Implementing External SCM Repositories 253**

Implementing a Subversion Repository	253
Prerequisites for Creating a Subversion Repository in Blaze Advisor.....	254
Supported Versions of Subversion Clients and Servers	254
Installing the SVNkit Plugin	255
Creating a Subversion Repository in Blaze Advisor	255
Creating a Subversion Repository Using the Trunk or a Branch	258
Creating a Subversion Repository Using the Utility	258
Writing a Connection Configuration File for a Subversion Repository	258
Writing a Repository Configuration File for a Subversion Repository	260
Writing a Batch File for Creating a Subversion Repository	262
Creating Workspaces or Workspace Connections for a Subversion Repository.....	263
Implementing a ClearCase Repository	263
Prerequisites for Creating a ClearCase Repository	264
IBM Rational ClearCase Jar Files	264
Creating a VOB	265
Creating a View on the IBM ClearCase Server	265
Creating a View For Each User	267
IBM Rational ClearCase Views and Blaze Advisor Workspaces	267
Creating a ClearCase Repository in the Blaze Advisor IDE.....	267

Creating a ClearCase Repository Using a Branch	270
Creating a ClearCase Repository Using the Utility	270
Writing the Connection Configuration File for a ClearCase Repository	270
Writing the Repository Configuration File for ClearCase Repository	273
Writing a Batch File for Creating a ClearCase Repository	274
Creating ClearCase Workspace Connections	275
Editing the Repository Configuration File	276
Creating RMA ClearCase Workspace Connections.....	277
Generating an RMA with a ClearCase Workspaces Directory Parameter	277
Creating an RMA Workspace Connection File	278
Changing the History View	279
Migrating an SCM Repository Using IBM Rational ClearCase 7.1	280
Prerequisites For Updating a Repository Configuration.....	280
Updating Your ClearCase Repository Configuration Using ClearTeam Explorer	281
Updating Your ClearCase Repository Configuration Using the NdRomAdminUtil Utility ..	281
Connecting to an SCM Generic Repository	282
Using the External SCM Versioning System Outside the Blaze Advisor Context	284

CHAPTER 9

Implementing a Database Repository..... **285**

Databases Supported by Blaze Advisor	286
Setting Your Classpath for the Database Driver Class	286
Setting a CLASSPATH Using a Command Prompt.....	286
Setting a CLASSPATH Using Blaze Advisor	286
Creating a Database Table	287
Running a Utility to Create a Database Table.....	287
Manually Creating a Database Table	289
Changing the Default Data Type for a Field	290
Performance and Tuning for Databases.....	290
Creating a Database Repository Using the Blaze Advisor IDE	291
Database Repository With no Versioning.....	291
Database Repository With a BVS Private Workspace	293
Creating a Database Repository Using the Utility	296
Creating a Database Repository With No Versioning Using the Utility.....	296
Writing a Connection Configuration File for a Database Repository	296
Writing a Repository Configuration File for a Database Repository	298
Writing a Batch to Create a Database Repository	299
Creating a Database BVS Repository With Private Workspaces Using the Utility	300
Writing a Connection Configuration File for a Database BVS Repository	301
Writing a Repository Configuration File for a Database BVS Repository	303
Writing a Batch File to Create a Database BVS Repository	304
Creating Workspaces or Workspace Connections for your Database Repository	305
Connecting to a Database Repository by Using an External Connection	306
Converting a Repository to and from a Database Repository	307

CHAPTER 10

Implementing MongoDB Repositories

309

MongoDB and Blaze Advisor Repositories	309
--	-----

Installation and Setup of the MongoDB Database.....	310
Versions Supported	310
Opening the MongoDB Port	311
Closing the Default Port	311
Installing the MongoDB Database	311
Creating the Data Folder	312
Creating a Log Folder	312
Configuring the MongoDB Database	313
Running the MongoDB Database	313
Running the MongoDB Database as a Service on Windows	314
MongoDB Authentication and Blaze Advisor Repositories.....	314
MongoDB URL Connection Parameter	315
MongoDB Authentication Type Connection Parameter	315
Connecting to MongoDB in Non-authentication Mode	316
Connecting to MongoDB in Authenticated Mode	316
Creating Authorized Users.....	317
Using the Localhost Exception	317
Using Administrative Credentials	319
Using the Mongo Shell	320
Creating an Administrative User	321
Blaze Advisor MongoDB Repository Configurations	322
Setting the Item Content Threshold	326
Creating a MongoDB Repository Using the Blaze Advisor IDE	327
Creating a MongoDB Repository (no versioning) using the Blaze Advisor IDE	328
Creating a MongoDB Repository (with Authorization Manager) using the Blaze Advisor IDE	329
Creating a MongoDB Repository (BVS versioning - private) using the Blaze Advisor IDE	331
Creating a MongoDB Repository (BVS versioning - shared) using the Blaze Advisor IDE	334
Creating a MongoDB Repository (BVS versioning - private local file workspace) using the Blaze Advisor IDE.....	335
Creating a MongoDB Repository Using the NdRomAdminUtil Utility	338
Creating a MongoDB Repository (no versioning) Using the NdRomAdminUtil Utility.....	339
System Connection Configuration File for the Admin Repository	340
MongoDB Repository (no versioning) Connection Configuration File	341
Repository Configuration File for a MongoDB Repository (no versioning)	343
Creating a Batch File or Running NdRomAdminUtil Utility to Create a MongoDB Repository 344	
Creating a MongoDB Repository (with Authorization manager) Using the NdRomAdminUtil utility.....	346
MongoDB Repository (with Authorization manager) Connection Configuration File	346
Repository Configuration File for a MongoDB Repository (with Authorization manager)	349
Creating a MongoDB Repository (BVS versioning - private) Using the NdRomAdminUtil utility..	350
MongoDB Repository (versioning - private) Connection Configuration File	351
Repository Configuration File for a MongoDB Repository (versioning - private)	353
Creating a MongoDB Repository (BVS versioning - shared) Using the NdRomAdminUtil utility..	355
MongoDB Repository (versioning - shared) Connection Configuration File	356
Repository Configuration File for a MongoDB Repository (versioning - shared)	358

Creating a MongoDB Repository (BVS versioning - private local file workspace) Using the NdRomAdminUtil utility.....	360
MongoDB Repository (BVS versioning - private local file workspace) Connection Configuration File	361
Repository Configuration File for a MongoDB Repository (BVS versioning - private local file)	363
Creating or Removing a MongoDB or File Workspace	365
Creating a Workspace for a MongoDB Repository (versioning - private) Using the Blaze Advisor IDE	365
Creating a Workspace for a MongoDB Repository (BVS versioning - private local file) Using the Blaze Advisor IDE	368
Creating a Workspace for a MongoDB Repository (versioning - private) Using the NdRomAdminUtil Utility	370
Creating a Workspace Connection Configuration File for the MongoDB Repository (versioning - private)	370
Creating a Batch File or Running the NdRomAdminUtil Utility to Create a Workspace for a MongoDB Repository (versioning - private)	373
Creating a Workspace for a MongoDB Repository (BVS versioning - private local file) Using the NdRomAdminUtil Utility.....	373
Creating a Workspace Connection Configuration File for a MongoDB Repository (BVS versioning - private local file)	374
Creating a Batch File or Running the NdRomAdminUtil Utility to Create a Workspace for a MongoDB Repository (BVS versioning - private local file)	376
Removing a MongoDB or File Workspace Using the NdRomAdminUtil Utility.....	376
Creating a Batch File or Running the NdRomAdminUtil Utility to Remove a MongoDB Workspace or a File Workspace Connected to a MongoDB Repository	377
Connecting to a MongoDB Repository Using the Blaze Advisor IDE	378
Creating a New Connection for a MongoDB or File Workspace or Repository Using the Blaze Advisor IDE.....	378
Viewing the Repository or Workspace in the Blaze Advisor IDE	381
Converting to a Cloud Compatible MongoDB Repository	381
Converting a MongoDB Repository (BVS versioning - private local file) to a MongoDB Repository (BVS versioning - private)	382
Regenerating an RMA After Converting to a MongoDB Repository	383
SSL Connections and MongoDB Repositories	383
Configuring MongoDB to use SSL	383
Testing MongoDB with a Self-signed SSL Certificate.....	384
RMA Generation for MongoDB Repositories and Workspaces	385
Workspace Base Name Setting for MongoDB Workspaces in the RMA Generator	386
CHAPTER 11	
Implementing an Authorization Manager.....	389
Writing a Custom Authorization Manager	390
Adding Import Statements	391
Implementing the Blaze Advisor Authorization Manager.....	391
Opening and Closing a Repository Connection	391
Handling Repository-level Authorization Requests	392
Repository-level Authorization Requests	392
Repository-level Error Messages	394
Handling Project-level Authorization Requests.....	394

Authorization Request Context	395
Project-level Authorization Requests	395
Project-level Error Messages	399
Connection Context and Authorization Requests	400
Setting Authorization Requests for Users.....	400
Setting Root and Directory-level Requests for Repository-level Authorization	400
General Guidelines When Setting Authorization Requests	401
General Guidelines When Setting Authorization Requests for RMAs	401
Limitations When Setting Authorization Requests	402
Creating a Repository With a Custom Authorization Manager Class	402
Creating a Repository With an Authorization Manager	402
Connecting to a Repository With an Authentication and Authorization Service	403
Connecting to a Repository With an Authorization Service	403
Connecting to a Repository With an Authorization Manager in an RMA	403
Connecting to a Repository With an Authorization Service in a Deployed Service.....	404
Addendum A: Setting Typical Categories of Permissions for Project-level Authorization	405
Addendum B: Setting Typical Categories of Permissions for Repository-level Authorization	409

CHAPTER 12**Implementing Content Protection for Repository Items 413**

About the IP Protection Framework	413
Code Access Security.....	414
Code Tampering	414
Limitations of the IP Protection Framework	414
Implementing the IP Protection Framework	414
Writing the IP Protection Classes.....	415
Obtaining Authenticated Identity Tokens	415
About Identity Tokens	416
Generating a Signature	416
Verifying a Signature	416
Obtaining Privileged Users and Encryption Keys	416
Identifying a Privileged User	417
Instantiating an Authenticated Identity Holder	418
Connecting the Authentication Service to the Repository.....	418
Configuring the Repository	418
Instantiating an Authentication Service	419
Initializing the IP Protection Framework	419
Encrypting and Decrypting Protected Item Content	419
Encrypting and Decrypting Content	419
Instantiating an Encryption Content Converter	420
Generating Signatures.....	420
Instantiating a Signature Generator	420

Index 423

CHAPTER 1

Designing a Blaze Advisor Repository

A key aspect of the FICO® Blaze Advisor® decision rules management system is the ability to create a repository to maintain and store your rules. Blaze Advisor supports three main default repository types, File and Database (RDBMS) and MongoDB (NoSQL).

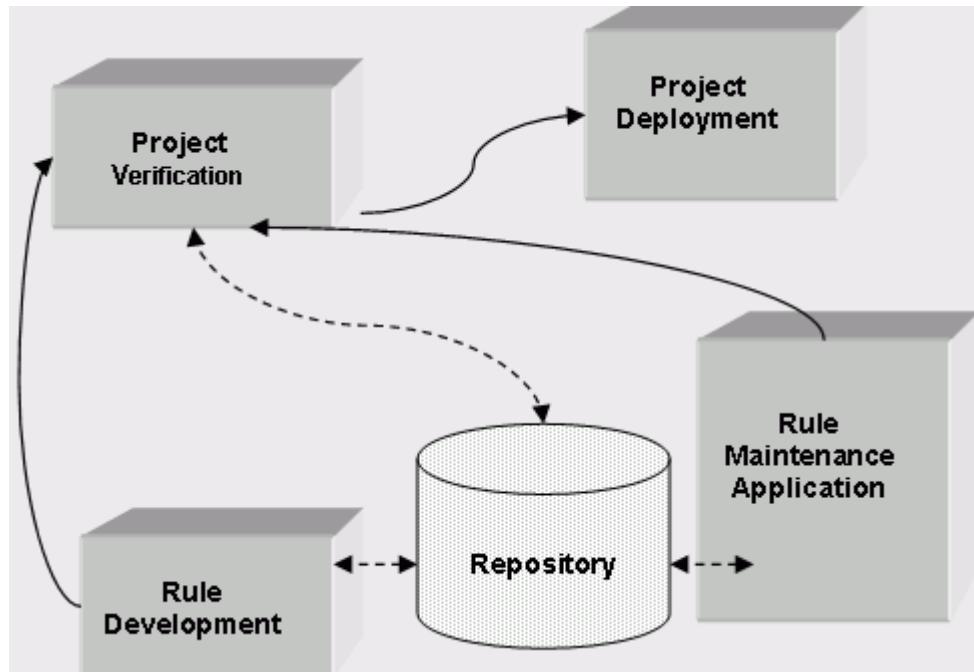
This section contains information that explains what you need to consider before choosing a Blaze Advisor repository persistence type related services.

- “[Blaze Advisor Repository Overview](#)” on page 13
- “[Key Decision Points for Choosing a Blaze Advisor Repository](#)” on page 14
- “[The Blaze Advisor Repositories and Workspaces](#)” on page 21
- “[Implementing Repositories](#)” on page 25

Blaze Advisor Repository Overview

A Blaze Advisor repository is a persistence mechanism that allows your developers to store, retrieve, and organize their directories and items. Your users can connect to a repository and work on these files in the Blaze Advisor IDE or a rule maintenance application (RMA). You can use a Blaze Advisor repository with your existing infrastructure components. For example, Blaze Advisor can be used with existing RDBMS or NoSQL databases or source control management systems.

This diagram shows the interactions between a Blaze Advisor repository and the other application components.



Before you can create a Blaze Advisor rule project or rule service, you need to create a repository. Blaze Advisor supports three default repository types, File and Database(RDBMS) and MongoDB (NoSQL). You can also add a source control or security services to your repository and save your workspace contents to a .zip file. If you are uncertain which repository type and services best suits the needs of your project, see ["Key Decision Points for Choosing a Blaze Advisor Repository" on page 14](#).

If you want to read more about the repositories and workspaces, see ["The Blaze Advisor Repositories and Workspaces" on page 21](#).

You can use either the Blaze Advisor IDE or a command-line utility to create the type of repository which best suits your project requirements. See ["Implementing Repositories" on page 25](#).

Key Decision Points for Choosing a Blaze Advisor Repository

There are several factors you need to consider when choosing a type of repository persistence for your Blaze Advisor project:

- ["Characteristics of Repository Types and Suggested Usage" on page 15](#)
- ["Source Control Management" on page 17](#)
- ["Security Considerations" on page 19](#)

After you have decided what type of repository you would like to create, see ["Implementing Repositories" on page 25](#).

Characteristics of Repository Types and Suggested Usage

Blaze Advisor supports three default repository types: File and Database (RDBMS) and MongoDB(NoSQL). The table below lists the characteristics of the each of the default repository types and their typical usage.

Repository Type	Characteristics and Typical Usage
File repository	<p>This repository type is the easiest to implement and use. Because the files reside on the local file system, it is possible to edit them manually.</p> <p>The File repository provides users with direct access to files. This can be an advantage if you have a small development team and frequent changes to the project files do not have to be made or tracked.</p> <p>In general, File repository is best suited for prototypes or for a project with low security and/or concurrency requirements.</p> <p>Note The Default Repository Connection that comes with your Blaze Advisor installation connects to a File repository. Blaze Advisor supports a built-in SCM system called Blaze Versioning (BVS) that can be used with File repositories with File workspaces. See “Source Control Management” on page 17.</p>
Database repository	<p>The Database repository provides additional security and protection for repository contents. Users must be granted the rights to access the database before they can view or modify the repository contents. The advanced tools for replication, backup, and recovery provided by the database are available to the repository.</p> <p>Administrators are required to set up and maintain the database system.</p> <p>Blaze Advisor supports two different database types. For a comparison, see “Deciding on a Database Type for your Repository” on page 16.</p> <p>Blaze Advisor supports a built-in SCM system called Blaze Versioning (BVS) that can be used with Database repositories with a private File workspace. See “Source Control Management” on page 17.</p> <p>Note The Database repository is best suited for projects with high security and backup protection requirements.</p>

Repository Type	Characteristics and Typical Usage
MongoDB repository	<p>The MongoDB repository provides a NoSQL or schemaless database option for users who are thinking of deploying an RMA or rule service in the cloud.</p> <p>The MongoDB repository type is available with and without Blaze Versioning (BVS) that supports a shared MongoDB workspace, private MongoDB workspaces, and private File workspaces and it can be configured to use an Authorization manager.</p> <p>Blaze Advisor supports two different database types. For a comparison, see “Deciding on a Database Type for your Repository” on page 16.</p> <p>For more information on these repository types, see “Source Control Management” on page 17.</p>
Repository with a source control management (SCM) system	<p>The addition of an SCM system prevents users from overwriting changes made by others and allows for revision tracking.</p> <p>Blaze Advisor supports a built-in SCM system called Blaze Versioning (BVS) for all of its default repository types and also supports external SCM systems including:</p> <ul style="list-style-type: none"> ■ File Repository (CVS versioning) ■ File Repository (Subversion versioning) ■ File Repository (ClearCase versioning) <p>It also supports File Repository (Generic SCM versioning). If you created a SCM repository using ClearCase, in an earlier version of Blaze Advisor, you can still create connections to workspaces for this repository. Note that the File Repository (ClearCase versioning) supports a later version of ClearCase.</p> <p>For more information on these repository types, see “Source Control Management” on page 17.</p> <p>Note A repository with an built-in or external SCM system is best suited for projects with medium security and significant software asset management requirements.</p>



Important You need to consider how the network connection and the repository type you choose for your project will impact performance, see “[Network Connection and Repository Type Considerations For Rule Service Deployment Performance](#)” on page 57.

Deciding on a Database Type for your Repository

If you want to store your repository in a database, you have two main options:

Database	MongoDB
Type: SQL/RDBMS	Type: NoSQL or Schemaless
Supported Vendors: <ul style="list-style-type: none"> ■ Hypersonic HSQL ■ Oracle ■ IBM DB2 ■ Microsoft 	Supported Vendors: <ul style="list-style-type: none"> 10gen

Database	MongoDB
Data structure for storage: Repository is stored a table	Data structure for storage: Repository is stored in a collection.
Source control management: Blaze Versioning (BVS) Available combinations: ■ Database Repository (no versioning) ■ Database Repository (BVS versioning - private) by default uses file workspaces. For more information see " Source Control Management " on page 17.	Source control management: Blaze Versioning (BVS) Available combinations: ■ MongoDB Repository (no versioning) ■ MongoDB Repository (with Authorization manager) ■ MongoDB Repository (BVS versioning - private) uses MongoDB workspaces ■ MongoDB Repository (BVS versioning - shared) ■ MongoDB Repository (BVS versioning - private local file workspace) For more information see " Source Control Management " on page 17.
Benefits: ■ Several vendors supported ■ Single BVS versioning type supported. ■ Easy conversion from a File repository to a Database repository. ■ RMA workspace type is File. ■ Not Cloud compliant	Benefits: ■ Single vendor supported. ■ Several BVS versioning types supported. ■ Easy conversion from a File repository to a MongoDB repository. ■ RMA workspace type is MongoDB if a MongoDB Repository (BVS versioning - private) is used. ■ RMA workspace type is File if a MongoDB Repository (BVS versioning - private local file workspace) is used. ■ Cloud compliant

Source Control Management

You may want to use a source control management system with your repository to track revisions of your project entities. Blaze Advisor supports several default versioning service options. This table lists the available default Blaze Advisor versioning services and their merits:

Versioning Option	Characteristics
<p>No versioning This option is acceptable if the project files are versioned externally or if tracking the revision history is not relevant for your project.</p>	<p>The project files are not versioned. Revision history information is not stored. For general information about workspaces and repositories, see “The Blaze Advisor Repositories and Workspaces” on page 21.</p>
<p>Blaze Versioning Service (BVS) This versioning option is often chosen when you want to use a source control management (SCM) system but do not want or do not have access to an external system. Most of the versioning commands you would typically use with an external SCM systems are available in a repository with BVS versioning.</p>	<p>BVS is a built-in versioning service that can be used with a File or Database repositories and is simple to implement. For more information about BVS workspaces, see “Deciding on the Type of BVS Workspace” on page 209. BVS provides two types of workspaces:<ul style="list-style-type: none">■ Shared (for File repositories) See “Shared Workspaces” on page 213.■ Private (for File and Database repositories) See “Private Workspaces” on page 211.For general information about workspaces and repositories, see “The Blaze Advisor Repositories and Workspaces” on page 21. By default, the private workspaces are of file type, however you can create a workspace that is of database type. See “Private Workspaces” on page 211. Branching and merging are not supported for BVS Versioning.</p>

Versioning Option	Characteristics
<p>External SCM System</p> <p>This versioning option is typically used when a external source control management system is already being used to manage other project components.</p>	<p>In addition to the built-in BVS versioning service, CVS, Subversion and ClearCase versioning can be used.</p> <ul style="list-style-type: none"> ■ The File Repository (CVS versioning) uses the JavaCVS client. This client is installed with Blaze Advisor. Once connected to the repository, users work in a local workspace. See “Local Workspaces” on page 238. Branching is supported for CVS versioning, however, you need to create the branch on the server before you can create a repository using the branch location. Merging is not supported. ■ The File Repository (ClearCase versioning) repository type supports IBM Rational ClearCase server connections. Once connected to the repository, users work in a local workspace. See “Local Workspaces” on page 238. Branching is supported for ClearCase versioning, however, you need to create the branch on the server before you can create a repository using the branch location. Merging is not supported. ■ The File Repository (Subversion versioning) repository type supports Subversion server connections. Once connected to the repository, users work in a local workspace. See “Local Workspaces” on page 238. Branching is supported for Subversion versioning. You need to create the branch on the server before you can create a repository using branch location. Merging is not supported. ■ The File Repository (Generic SCM versioning) is available for backward compatibility. If you created an SCM repository connecting to a ClearCase server in an earlier version of Blaze Advisor, you can use this repository type to create connections to existing workspaces for those SCM repositories. Note that the File Repository (ClearCase versioning) in this release supports a later version of ClearCase. <p>For general information about workspaces and repositories, see “The Blaze Advisor Repositories and Workspaces” on page 21.</p>

Security Considerations

Most Blaze Advisor default repository types can be configured to use authentication or a custom authorization manager class.

- “[Blaze Advisor Authentication](#)” on page 19
- “[Blaze Advisor Authorization](#)” on page 20
- “[Blaze Advisor Content At Rest Management](#)” on page 21

Blaze Advisor Authentication

Authentication is the process of determining if a user or entity is who they claim to be and serves as the foundation for enforcement of subsequent access control policies.

Blaze Advisor recommends that you implement authentication for users who need to access your rule project, rule maintenance application, or deployed rule service according to your existing corporate authentication policies and procedures.

Blaze Advisor repository authentication is usually handled externally, for example through the use of a third-party LDAP-based system.

The Single Sign-on and Embedding RMA Example included with your Blaze Advisor installation demonstrates how you can implement a Single Sign-on (SSO) environment for accessing RMAs. After authenticating a user upon sign-on through an LDAP server, the user credentials are stored as session attributes, so users are not required to enter their credentials through the sign-on screen for each application. See the “Single Sign-on and Embedding an RMA Example” on page 79 of *Examples.pdf*. See the Single Sign-on and Embedding an RMA Example in the FICO Blaze Advisor Help.

Blaze Advisor Authorization

User authorization is the process of controlling *what* actions authenticated users can perform on *which* repository entities. Blaze Advisor recommends that you apply your organization’s best practices for authorizing users to areas of the rule project, rule maintenance application, or deployed rule service as appropriate.

For example you can use the following approach to identify the users, tasks, resources, and constraints:

- Identify the **resources** that need to be protected. Examples of Blaze Advisor resources include repository entities such as business objects, rules and rulesets, decision metaphors, and template and instances.
- Identify the **actions/operations** that can and/or cannot be performed over the Blaze Advisor resources. Typical operations include reading, writing, and creating a new instance, and modifying or deleting an existing instance.
- Identify the **roles** in the application. These typically can be obtained from the use-case scenarios for the application where they are often represented as the actors or subjects. It is important at this stage to identify hierarchies as well. Applications must be coded to ensure users can be segmented by role with appropriate authorization/privilege level assignments (Separation of Duties). For a description of some typical Blaze Advisor roles, see “Roles and Responsibilities” in *BestPractices.pdf*.
- Identify the **authorization constraints**. This step aims to bring the **resources**, the **actions/operations** that may be performed on those resources, and the **roles** together to define which roles can perform what operations. For instance, you may want to limit your users with the Business Analyst role to only editing instances in the RMA whereas you may want your Rule Writer to have access to both the repository entities in the Blaze Advisor IDE as well as the instances in the RMA.

Blaze Advisor provides a Authorization Manager interface that can be used to implement authorization services at the repository that can be used in the Blaze Advisor IDE and an RMA. This Authorization Manager interface can be written to check the login credentials stored in different types of files, systems, or services to authenticate and authorize users.

The Blaze Advisor Authorization Manager framework is the proper mechanism for registering an external system or building your own authorization system. The Authorization Manager can be activated for any type of operation (create, modify, delete, check out/in, promote, list...) on any entity by any user.

The Authorization Manager is registered at the repository level and therefore applies to all connections including the Blaze Advisor IDE for the technical users such as the Rule Writer and rule maintenance application for the business users such as the Business Analyst. See Authorization Manager Example in the FICO Blaze Advisor Help.

For fine-grained authorization management, the ideal solution is to manage the user/group authorization in an external system such as LDAP or database. Administrators can grant access to the entities in the repository as appropriate. A synchronization mechanism must take place in order to reflect the content of the repository in the external system.

In order to reduce the administration tasks, authorization management can also be implemented as a routine based on groups. For example users can edit anything that is owned by their group but have read-only or no access to rules owned by other groups. This mechanism can be delegated to an external system and be tailored to operate on management properties. As of today, authorization manager implementations typically grant and restrict access per folder and/or groups of files as file-level authorization management is viewed as incurring too much administrative overhead.

If you want to provide authorization management at the rule maintenance application mainly for your business users without configuring an Authorization Manager at the repository-level, see "Managing Instances Using Life Cycle Management" in *DevelopingRuleMaintenanceApplications.pdf*.

Blaze Advisor Content At Rest Management

Protection of critical organizational assets is best done in a layered manner ensuring that if one security layer/mechanism is compromised that the trespasser immediately encounters another "wall." For many organizations the business logic encapsulated in its business rules represent a company's critical competitive market advantage and, as such, need to be protected at all costs.

Under such circumstances the Blaze Advisor converter facility can be used to encrypt repository content thereby preventing someone who has gained unauthorized access to the server that physically hosts the repository from understanding the content. See "[Implementing Content Protection for Repository Items](#)" on page 413.

The Blaze Advisor Repositories and Workspaces

In the Blaze Advisor IDE, you can open or import a project using an existing Blaze Advisor repository or workspace connection to work with its project items in an Eclipse workspace. When you import a Blaze Advisor project into an Eclipse workspace you see the project and project items in the Project Explorer. The Eclipse workspace contains project references to files stored in a Blaze Advisor workspace or repository. You can add, edit and remove directories, project files, and items using Blaze Advisor and save the changes to the Blaze Advisor repository or workspace via the Eclipse workspace. The

commands you have available to you are based on whether or not you have a project open in the Blaze Advisor perspective. The contents of the Eclipse workspace can differ from what is stored in the repository or workspace depending on whether or not you have refreshed the contents of a project stored in a non-versioned repository or updated the contents of a project stored in a versioned repository.

There are also differences between the contents and physical location of a workspace and its repository depending on whether or not you have created a versioned or non-versioned repository:

- If you have created a repository of any type that does not use a versioning service, the Blaze Advisor repository or workspace contents are *logically identical*. This means that when you refresh your workspace or repository, the project entities are the same as those in the Eclipse workspace.
- If you have created a Blaze Advisor File repository with a BVS shared workspace, the Blaze Advisor workspace and the Blaze Advisor repository share the same physical location and the Eclipse workspace references projects and project items in this physical location. But you need to refresh the Eclipse workspace and/or check in any entities that you have changed for the content in the Eclipse workspace and the repository contents to be logically identical.
- If you have created a versioned Blaze Advisor repository with a private or local workspace and imported any projects to the Eclipse workspace, the Eclipse workspace contains references to the project and project files in the Blaze Advisor workspace which has a connection to the Blaze Advisor repository. The Blaze Advisor workspace and repository do not share the same physical location and may or may not contain contents that are logically identical, unless you check in any changes or update the contents of the Eclipse workspace. The Blaze Advisor repository and the workspace are considered separate objects.

For more information about workspaces in a versioned repository:

- ["Deciding on the Type of BVS Workspace" on page 209](#)
 - ["Private Workspaces" on page 211](#)
 - ["Shared Workspaces" on page 213](#)
 - ["Local Workspaces" on page 238](#)

For information about RMA workspaces, see ["Connecting to a Blaze Advisor Repository From a Rule Maintenance Application" on page 55](#).

The Repository Structure

When you launch Eclipse and select a workspace, you can open projects using any existing Blaze Advisor workspace or repository connections. You select an existing workspace or repository connection from the list when you want to display the contents of the workspace or repository in the Repository Explorer or when you want to import a Blaze Advisor project into the Eclipse workspace using the Import wizard. If you want to display the contents of an existing workspace or repository using the Repository Explorer,

You see the repository icon and the workspace or repository connection name you specified when you created the connection using the Blaze Advisor IDE or the `NdRomAdminUtil` utility. When you expand this top directory for an existing workspace or repository, you see the rest of the workspace or repository contents displayed in a tree structure. If you view the contents of a new repository in the Repository Explorer, the only entity you see is the repository icon.

Whether you select a project from the Repository Explorer or the import a project using the Import wizard, you see the project and its project files in the Project Explorer. In the Project Explorer the Repository name and the path to the project is displayed next to the project name. If you are using a repository with no versioning you can refresh your Project Explorer or Repository Explorer to see the files in the repository. If you are using a versioned repository with a local or private workspace, you would need to update your project to see the latest version of all files.

The workspace or repository entities belong to three main overlapping categories: entry, directory, and item. These terms are used in the Blaze Advisor API and the Blaze Advisor IDE. See ["Repository Entities" on page 24](#).

The repository structure you create for your project(s) may or may not resemble the repositories that are installed with Blaze Advisor however, these repositories represent examples of how we recommend that a repository, its directories and items be structured. See ["Repository Content Organization Practices" in BestPractices.pdf](#).

For information about the structure of shared, private, and local workspaces see:

- ["File Structure for a Private or Local Workspace" on page 212](#)
- ["File Structure for a BVS Shared Workspace" on page 214](#)

If you are new to Blaze Advisor and you would like some hands-on experience building a rule project in a repository, see ["Tutorials Overview" in Tutorials.pdf](#).

Storage Names and Names in the Blaze Advisor IDE

The names for the files and folders in a Blaze Advisor project that you see on the storage system, such as a file system or a database, are considered the *storage names*. For example, when you first create your folders and items in a project, you enter names for these entities and save their contents. If you are using the Windows Operating System, when you navigate to the directory location in Windows Explorer where your folders and items are stored, you see that the storage name and the name you gave these entries in Blaze Advisor are the same. If at some point in the future, you change the name of any of these folders and items in Blaze Advisor and save your changes, the name in the Blaze Advisor IDE is no longer the same as the storage name on your system. The storage name does not change because it is used by Blaze Advisor to perform a number of operations and therefore must be immutable. If you will be writing any custom classes or any SRL that reference any specific folders or items, you need to use the storage names.

Repository Entities

The following terms describe the types of entities in a Blaze Advisor repository and the differences between how the terms are used in the Blaze Advisor API and the Blaze Advisor IDE:

- Entry

There is a subtle distinction between the way the term "Entry" is used in the Blaze Advisor IDE and in the API. In the Blaze Advisor IDE the term "Entry" or "Entries" does not refer to directories or projects and is limited only to items.

- In the API, when the class or method refers to the *Entry* it is used to described a self-contained Blaze Advisor entity stored in a repository. An entry can be either a directory (folder) or an item.
- In the Blaze Advisor IDE, the term Entry is used to describe a subset of the above definition. When the term Entry is used in an editor such as the Filter Instance Editor, Management Properties Organizer, or the Query Instance Editor, it refers *any Entry* such as those in the Type drop-down list of the editor or organizer. In the Blaze Advisor IDE to be considered an entry, these items must be stored at the top-level for their information to be retrieved, meaning that they must be stored in a directory instead of being added locally to a Group template.

- Project

The project is considered a special type of item because it can contain references to other entities such as directories or subprojects that can contain other entries.

- Directory

A *directory* refers to entities that can contain other entries.

When you connect to a Blaze Advisor repository, you see the repository directory represented by a repository icon that is displayed at the top of the tree structure in the Repository Explorer.

- Item

An *item* describes entities including classes, rulesets, functions, decision metaphors, ruleflows, queries, verification instances, filters, templates, and instances that are considered terminal or end nodes. When we use the term *files* or *project items*, we are often referring to these items.

Repository Administration

During project development you may need to perform the following management tasks at the repository level:

- Adding management properties to a workspace or repository

Management Properties provide additional information about items in your project. For example, if you want to know when an item was created, you add a Creation Date management property. You add a management property to the repository so

that any projects or project items you create in that repository can use those properties. To avoid any system conflicts, we recommend that you add management properties to a repository immediately after you create it. Management property values are displayed in the Properties tab of an editor.

See “[Managing Repository Configuration and Connection Instances in the Admin Repository](#)” on page 131.

- Managing your project lifecycle

You can perform the following tasks importing or exporting workspaces or projects, releasing projects to a directory in the same repository, and publishing projects to another repository:

- “Importing Blaze Advisor Projects” in *DevelopingRuleProjects.pdf*
- “Releasing Blaze Advisor Projects” in *DevelopingRuleProjects.pdf*
- “Publishing Blaze Advisor Projects” in *DevelopingRuleProjects.pdf*
- “Exporting Blaze Advisor Projects” in *DevelopingRuleProjects.pdf*

- Retrieving information and performing administrative tasks on repository entities

You can use the `NdRomAdminUtil` utility to retrieve information about repository entities.

See “[Importing or Exporting Projects and Workspace Contents Using the Utility](#)” on page 65.

Implementing Repositories

If you want to know more about a specific default repository type or you are ready to create a repository, see the table below for links to specific topics for creating a repository using the Blaze Advisor IDE or command-line utility.



Note If you want to use a File repository (CVS versioning), File repository (ClearCase versioning), File repository (Subversion versioning), or a Database repository, you must complete some implementation tasks *prior* to creating a repository.

Repository Type	For more information
File repository	See “ Implementing a Blaze Advisor Repository ” on page 27.
File repository (BVS versioning - private)	See “ Implementing a Repository With a Blaze Versioning Service ” on page 209.
File Repository (BVS versioning - shared)	
File Repository (CVS versioning)	See “ Implementing a CVS Repository ” on page 235. CVS repositories use a local File workspace.
File Repository (Subversion versioning)	See “ Implementing a Subversion Repository ” on page 253. Subversion repositories use a local File workspace.

Repository Type	For more information
File Repository (ClearCase versioning)	See “ Implementing a ClearCase Repository ” on page 263 . ClearCase repositories use a local File workspace.
File Repository (Generic SCM versioning)	See “ Connecting to an SCM Generic Repository ” on page 282 . Generic SCM repositories use a local File workspace.
Database Repository (no versioning)	See “ Implementing a Database Repository ” on page 285 .
Database Repository (BVS versioning - private)	
MongoDB Repository	See “ Implementing MongoDB Repositories ” on page 309 .
MongoDB Repository (no versioning)	
MongoDB Repository (with Authorization Manager)	
MongoDB Repository (BVS versioning - private)	
MongoDB Repository (BVS versioning - shared)	
MongoDB Repository (BVS versioning - private local file workspace)	

CHAPTER 2

Implementing a Blaze Advisor Repository

You can create a FICO® Blaze Advisor® decision rules management system repository and configure it with the services required by your development projects. The persistence types supported by Blaze Advisor include File Database (RDBMS), and MongoDB(NoSQL). These repository types can be configured with services such as versioning or authentication and authorization. These repository services control how your users interact with the repository as they build or test the project and project items.

This section includes these topics:

- ["About Creating a Blaze Advisor Workspace or Repository" on page 27](#)
- ["Creating a Blaze Advisor Repository" on page 28](#)
- ["Creating a Local or Private Workspace" on page 226](#)
- ["Blaze Advisor Repository Contents and File Structure" on page 37](#)
- ["Converting a Repository to Use Another Type" on page 50](#)

If you want to use a Database, MongoDB, CVS, Subversion, or IBM Rational ClearCase server to host your repository, you need to complete some preliminary tasks discussed in one of the chapters below prior to creating a repository:

- ["Implementing a Database Repository" on page 285](#)
- ["Implementing a CVS Repository" on page 235](#)
- ["Implementing External SCM Repositories" on page 253](#)
- ["Implementing MongoDB Repositories" on page 309](#)

About Creating a Blaze Advisor Workspace or Repository

There are two ways to create a repository in Blaze Advisor. You can use the New Repository command in the Blaze Advisor IDE or you can use the `NdRomAdminUtil` utility. See ["Creating a Blaze Advisor Repository" on page 28](#).

After you create a repository, configuration files and a system folder are added to the root directory, see ["Blaze Advisor Repository Contents and File Structure" on page 37](#).

When you create a repository, you also create a workspace and a repository connection at the same time, if you are using a versioned repository with local or private workspaces, and you need to create additional workspaces, you can use the New Workspace

command in the Blaze Advisor IDE or you can use the `NdRomAdminUtil` utility. See “[Creating a Local or Private Workspace](#)” on page 226.

If you want to provide privileged users with the ability to set protections on repository entities, see “[Implementing Content Protection for Repository Items](#)” on page 413.

if you need to convert the repository type, see “[Converting a Repository to Use Another Type](#)” on page 50.

Creating a Blaze Advisor Repository

There are two methods you can use to create a Blaze Advisor Repository. The method you choose is based on your personal preference. The following topics provide general procedures for creating a repository.

- “[Creating a Repository Using the Blaze Advisor IDE](#)” on page 28
- “[Creating a Blaze Advisor Repository Using the Utility](#)” on page 33

For information on creating the following repositories. See:

- “[Implementing a Repository With a Blaze Versioning Service](#)” on page 209
- “[Implementing a CVS Repository](#)” on page 235
- “[Implementing a Database Repository](#)” on page 285
- “[Implementing External SCM Repositories](#)” on page 253
- “[Implementing MongoDB Repositories](#)” on page 309

Creating a Repository Using the Blaze Advisor IDE

You launch the wizard using the New Repository command from the Repository menu.

If you are creating a versioned repository with a private or local workspace using the Blaze Advisor IDE, you can specify the location of a workspace at the same time you specify the location of the repository directory and a repository connection.

If you are creating a workspace for an existing versioned repository, see “[Creating a Blaze Advisor Repository Using the Utility](#)” on page 33.

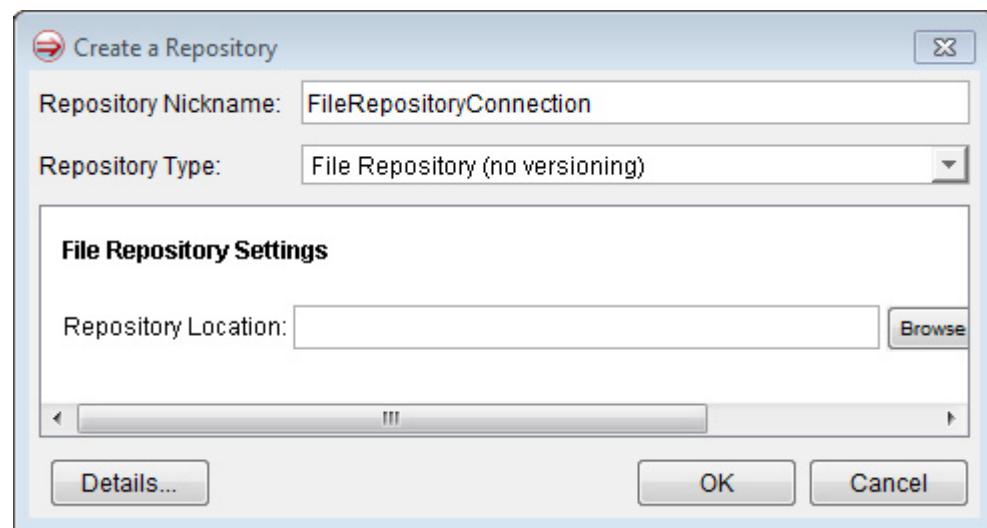
To create a repository

1 Complete any preliminary implementation tasks if you are planning to use a Database, MongoDB, Subversion, ClearCase or CVS server to host your repository.

2 Choose **Repository > New Repository**.

Alternatively, from the **File > New** and select **Repository** or in the Repository Explorer, select **New > Other** and expand the **Blaze Advisor** folder to choose **New Repository**.

The Create a Repository wizard opens.

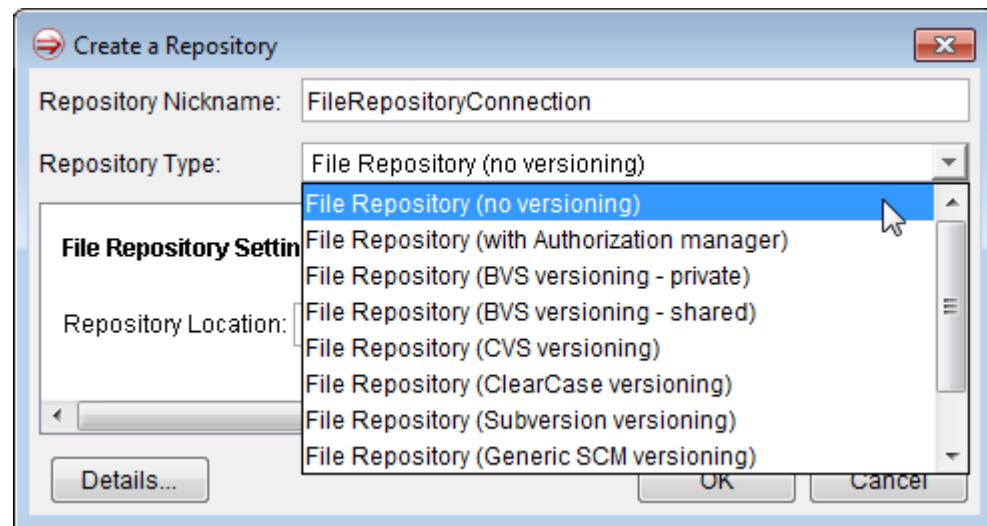


- 3 In the **Repository Nickname** field, enter a connection name for the repository you want to create.

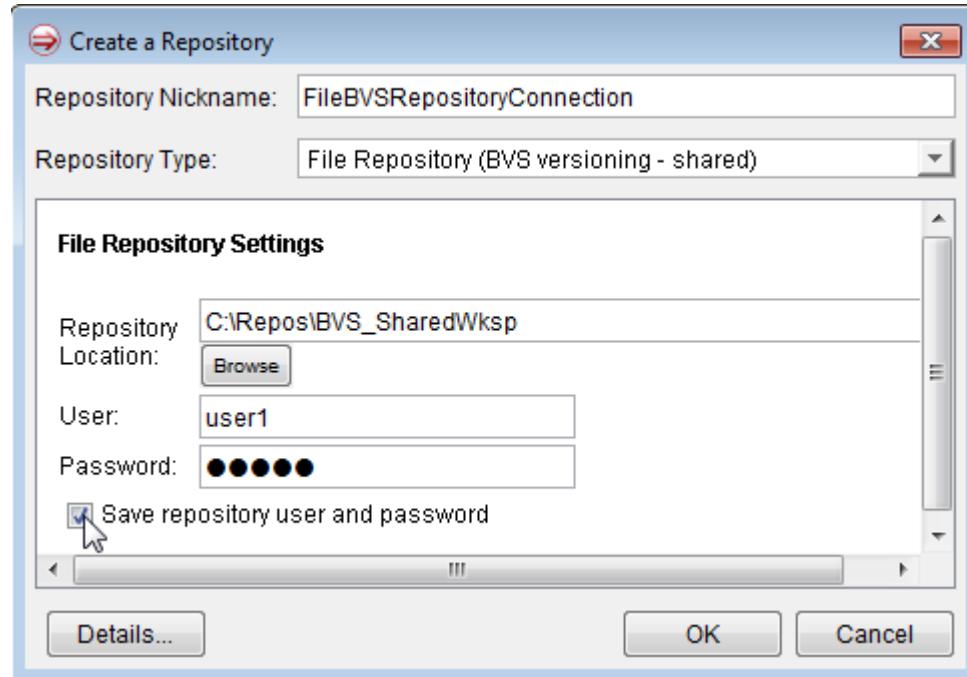
The name you enter here is displayed in the list of workspace and repository connections in the Manage Connections, Import, and Export wizards.

- 4 Choose an existing configuration from the **Repository Type** drop-down list.

The repository type you choose from the drop-down list is based on a pre-defined configuration.

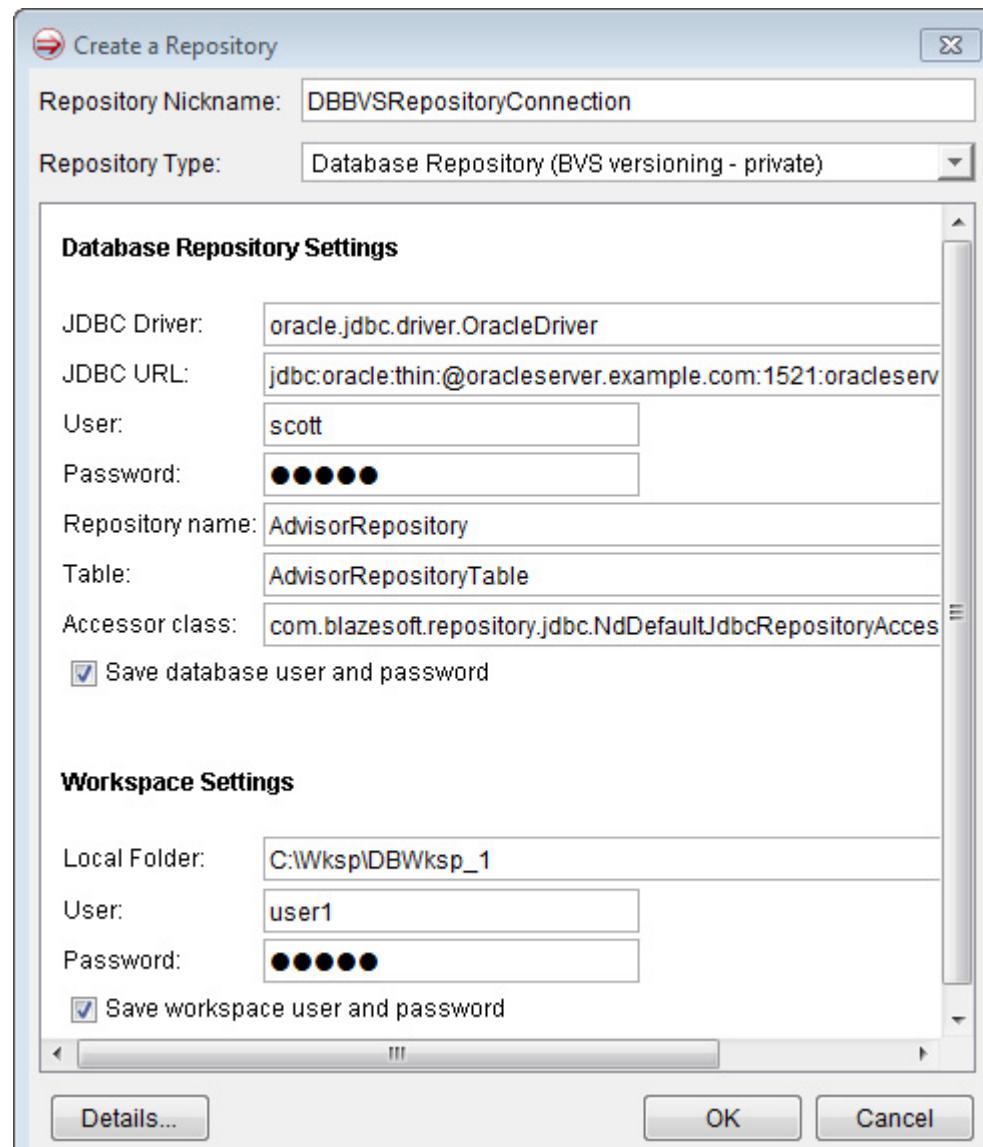


- 5 Enter the connection parameters for the type of repository you selected.
If you want to create a CVS repository with a connection to a CVSNT server, you need to use the **NdRomAdminUtil** utility. See "[Creating a CVSNT Repository](#)" on page 248.
The repository type you selected from the drop-down list determines the parameters you need to enter to connect to the repository.
- 6 (Optional) Depending on the repository type you select, you may also need to enter both a user name and password for the repository and the workspace.
- 7 (Optional) You can unselect the **Save user and password** check box.
 - This option is selected by default so that users can quickly and easily connect to the workspace, repository, or both by selecting a connection in the Manage Connections, Import, or Export wizards. The password is encrypted by default. See "[Saving User Name and Passwords](#)" on page 43.
 - If the option is unselected, user names and passwords are not persisted in the system, however it also means that users will be prompted whenever a connection to the workspace, repository, or both needs to be made.

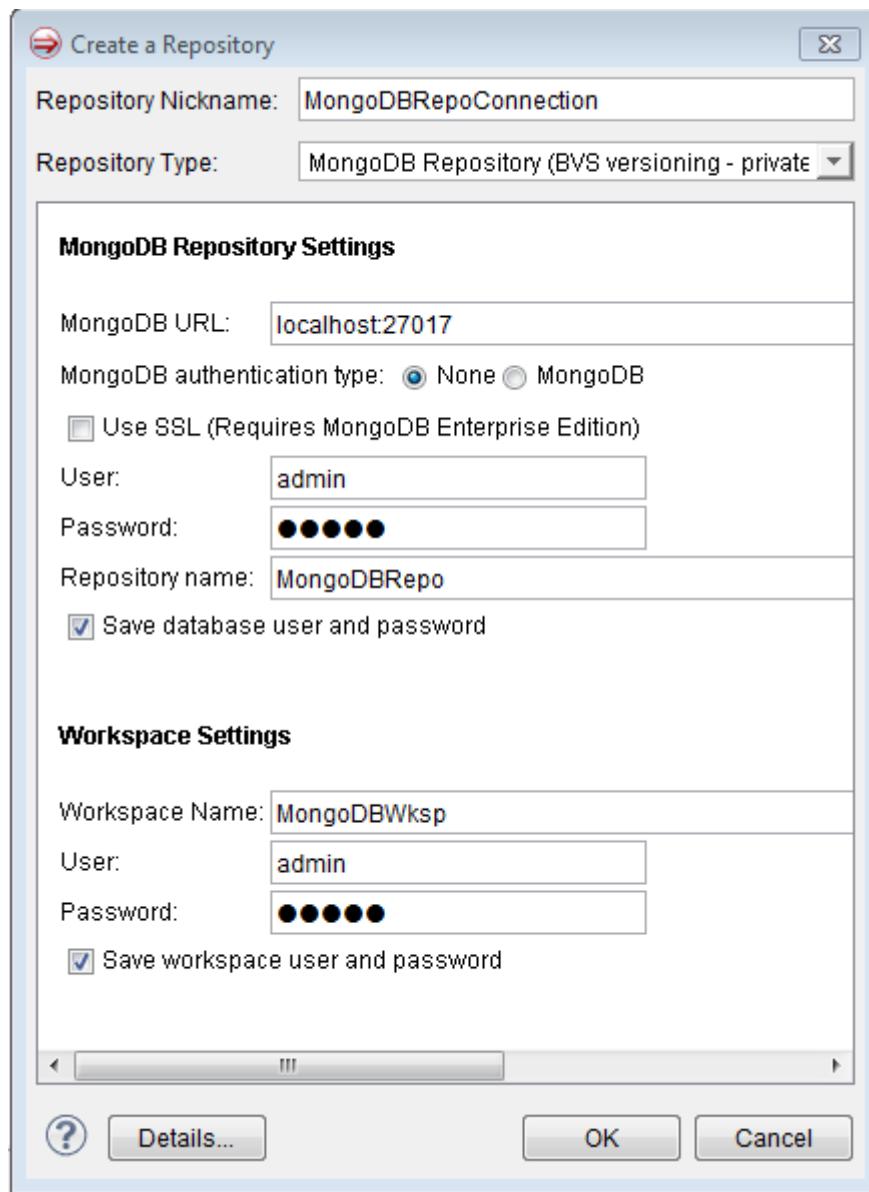


- 8 (Optional) Click **Details** to see the configuration parameters of the repository type you selected.
If you view the Details page and want to edit the configuration to add or delete a service, see "[Editing a Repository Configuration in the Blaze Advisor IDE](#)" on page 39.
- 9 (Optional) If you are creating a Database Repository with BVS private workspaces and want your users to enter workspace connection credentials that are different from

your repository server connection credentials, click **Details** button. See “[Allowing Distinct Repository and Workspace Users](#)” on page 49.



- 10 (Optional) If you are creating a MongoDB Repository (BVS versioning - private) or another MongoDB repository type and you want to set the threshold for the item's content, click **Details** button. See “[Setting the Item Content Threshold](#)” on page 326.



- 11 After you have completed entering the connection parameters, click one of the following items:

- **OK** to exit the wizard.
- **Cancel** to exit the wizard without creating a repository.

After you create the repository, proceed to the following tasks:

- Use the Repository Explorer to begin creating the overall repository structure with folders and projects.
- If you need to create a new workspace, see "["Creating a Local or Private Workspace" on page 226](#).

- If you need to create a new connection, see "[Creating a New Blaze Advisor Workspace or Repository Connection](#)" on page 58.

Creating a Blaze Advisor Repository Using the Utility

You can create one of the default repositories types supported by Blaze Advisor such as File, or Database repository with or without services using the `NdRomAdminUtil` utility.

Before you can run the utility, you need to write one or more connection configuration files and a configuration file:

- ["About Writing Files for Creating a Blaze Advisor Repository or Workspace Using the Utility"](#) on page 168
- ["Writing the Connection Configuration File for an Admin Repository"](#) on page 33
- ["Writing the Connection Configuration File for a File Repository"](#) on page 34
- ["Writing the Repository Configuration File for a File Repository"](#) on page 34
- ["Writing a Batch File to Create the Repository Using the Utility"](#) on page 35

After you create the repository, you can use some of these files to perform other tasks using the `NdRomAdminUtil` utility, see "[The NdRomAdminUtil Utility Commands](#)" on page 162.

If you are creating a file BVS repository with shared workspace or a File, Database or MongoDB repository with private workspaces or a CVS, Subversion, or Clearcase repository with a local workspace and you want to save your user name and password, see "[Saving User Names and Passwords Using the Utility](#)" on page 183.

If you are creating a file BVS repository with shared workspace or a File, Database or MongoDB repository with private workspaces or a CVS, or Subversion with a local workspace and you want to create a workspace, see "[Creating a Local or Private Workspace](#)" on page 226.

If you are creating a Database or MongoDB BVS repository with private workspaces and want to set the option for allowing distinct repository and workspace users, see "[Allowing Distinct Repository and Workspace Users When Using the Utility](#)" on page 184.

After creating the repository, you may need to perform some administrative tasks using the utility. To perform some of these tasks you need to reference the repository connection configuration file. See "[Importing or Exporting Projects and Workspace Contents Using the Utility](#)" on page 65.

Writing the Connection Configuration File for an Admin Repository

You need to create a connection configuration file for the Admin Repository so that a copy of the system folder in the Admin Repository and a reference to the Admin Repository location is added to the repository. When you create a repository using the Blaze Advisor IDE, the system folder and reference is added automatically.

The Admin Repository connection configuration file can be used when creating any of the types supported by Blaze Advisor. After you write this file, you can reuse the file for a

number of tasks you perform with the utility including creating a workspace or removing a repository.

You can create the file using a text editor and save the file with a .cfg extension. For information about the tags and values in the following example, see ["Repository Connection Configuration File Examples" on page 171](#).

Example Admin Repository connection configuration file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<RepositoryConnection>
    <Factory>
        com.blazesoft.repository.file.NdFileRepositoryConnection
    </Factory>
    <RepositoryFolder>C:\Blaze\Advisor\lib\Admin Repository
    </RepositoryFolder>
    <User>admin </User>
    <RepositoryName> Admin Repository </RepositoryName>
</RepositoryConnection>
```

Writing the Connection Configuration File for a File Repository

You write a repository connection configuration file with information including the connection class, the location of the repository directory, the repository connection name, and the repository name. This is the same information you would enter in the New Repository wizard in the Blaze Advisor IDE.

You can create the file using a text editor and save the file with a .cfg extension. For information about the tags and values in the following example, see ["Repository Connection Configuration File Examples" on page 171](#).

The following is an example of a repository connection configuration file for a File Repository:

```
<?xml version="1.0" encoding="UTF-8" ?>
<RepositoryConnection>
    <Factory>
        com.blazesoft.repository.file.NdFileRepositoryConnection
    </Factory>
    <RepositoryFolder>C:\Blaze\Advisor\lib\File Repository345</
RepositoryFolder>
    <RepositoryName> File Repository345</RepositoryName>
</RepositoryConnection>
```

Writing the Repository Configuration File for a File Repository

You write a repository configuration file with tags and values for names of classes for repository functionality. When you create the repository using the New Repository wizard in the Blaze Advisor IDE, you can use the Details page to view or edit repository configuration. After you have created the repository, if you want to edit the resulting repository file to add or remove services, you need to manually edit its contents. For more information:

- ["Blaze Advisor Repository Contents and File Structure" on page 37](#)

- “[Editing a Repository Configuration in the Blaze Advisor IDE](#)” on page 39

You can create the file using a text editor and save the file with a .cfg extension. For information about the tags and values in the following example, see “[Writing a Repository Configuration File](#)” on page 191.

The following is an example of a repository configuration file for a File Repository:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RomConfig>
    <AllowUnusedValues> true </AllowUnusedValues>
    <RomConnectionManagerConfig>
        <ConnectionMode> 0 </ConnectionMode>
        <RomConnectionManagerFactory>
            <JavaName>
                com.blazesoftware.template.repository.impl.NdDefaultRomConnection
                Manager
            </JavaName>
        </RomConnectionManagerFactory>
    </RomConnectionManagerConfig>
    <RomQueryManagerConfig>
        <RomQueryManagerFactory>
            <JavaName>
                com.blazesoftware.template.repository.query.NdRomDefaultQuery
                Manager
            </JavaName>
        </RomQueryManagerFactory>
    </RomQueryManagerConfig>
    <RomSchemaManagerConfig>
        <RomSchemaManagerFactory>
            <JavaName>
                com.blazesoftware.template.repository.impl.NdDefaultRomSchema
                Manager
            </JavaName>
        </RomSchemaManagerFactory>
    </RomSchemaManagerConfig>
</RomConfig>
```

Writing a Batch File to Create the Repository Using the Utility

After writing these three XML files, we recommend that you write a batch file to run the NdRomAdminUtil utility. See “[The NdRomAdminUtil Utility](#)” on page 161.

You reference the path to these files as argument values. See “[Running the NdRomAdminUtil Utility](#)” on page 166.

Command, Argument, or Option	Description
createRepository	Command to create a repository
-repositoryConnection	Argument takes the path to the location of the repository connection configuration file (.cfg). See "Repository Connection Configuration File Examples" on page 171 . This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
Note Use this argument to create a workspace for a BVS Repository with private workspaces, File CVS Repository File Repository (Subversion versioning) or a File Repository (ClearCase versioning) using an IBM Rational ClearCase server connection.	Argument takes the path to the location of the workspace connection configuration file (.cfg) See "Writing a Connection Configuration File for a Private or a Local Workspace" on page 185 . This file contains information such as the repository name, the workspace connection name, the user name and password for the connection, and the path to the workspace.
-systemconnection	Argument takes the path to the location of the system Folder connection configuration file (.cfg). See "Repository Connection Configuration File Examples" on page 171 . This file contains information such as the name of the repository which in this case is Admin Repository, the location of the Admin Repository and the user name.
-configuration	Argument takes the path to the location of the configuration file(.cfg). See "Writing a Repository Configuration File" on page 191 .
-verbose	Option prints out all details messages into the command console.
-m	Option allows users to provide a comment. Note Use double quotes around your comment text.
-debug	Option prints out tracing information into the command console.

Example of a batch file to create a File repository using pathnames:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil  
createRepository  
-repositoryConnection C:/Repo_Connections/ConnectionFiles/DefaultRepo.cfg  
-systemConnection C:/Repo_Connections/ConnectionFiles/sysAdmin.cfg -  
configuration C:/Blaze/Advisor/lib/NoVerGeneric.cfg -m "File Repository  
Created" -verbose
```

Example of a batch file to create a Database repository using pathnames:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil  
createRepository  
-repositoryConnection "c:/Repo_Connections/ConnectionFiles/  
DefaultJdbcRepo.cfg"  
-systemConnection "c:/Repo_Connections/ConnectionFiles/sysAdmin.cfg" -  
configuration "c:/Blaze/Advisor/lib/NoVerGeneric.cfg" -m "DB Repository  
Created" -verbose
```

Removing a Blaze Advisor Repository Using the Utility

You can remove a File Database repository with or without services using the NdRomAdminUtil utility. To remove the repository using the utility, you use the connection and configuration files that you used with you created the repository.

- “[Writing the Connection Configuration File for an Admin Repository](#)” on page 33
- “[Writing the Connection Configuration File for a File Repository](#)” on page 34
- “[Writing the Repository Configuration File for a File Repository](#)” on page 34
- “[Writing a Batch File for Removing a File Repository](#)” on page 37

Writing a Batch File for Removing a File Repository

In the batch file or command prompt, you reference the path to these files as argument values. See “[Running the NdRomAdminUtil Utility](#)” on page 166.

Command, Argument, or Option	Description
removeRepository	Command to remove a repository.
-repositoryConnection	Argument takes the path to the location of the repository connection configuration file (.cfg). See “ Repository Connection Configuration File Examples ” on page 171. This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
-verbose	Option prints out all details messages into the command console.

Example of a batch file to remove a File repository using pathnames:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil
removeRepository
-repositoryConnection "c:/Repo_Connections/ConnectionFiles/DefaultRepo.cfg"
-verbose
```

Example of a batch file to remove a Database repository using pathnames:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil
removeRepository
-repositoryConnection "c:/Repo_Connections/ConnectionFiles/
DefaultJdbcRepo.cfg"
-verbose
```

Blaze Advisor Repository Contents and File Structure

When you create a repository, two configuration files, three attribute (.attbs) files, and a system Folder are added to the root directory of your repository.

Repository and ROM Configuration Files

When you create a repository, the com.blazesoft.repository_config.cfg file and the com.blazesoft.rom_config.cfg file and a related.attbs file for each, are added to the

root directory of your repository. The two configuration files contain the configuration and connection parameters for your repository.

- The `com.blazesoftware.rom.config.cfg` file contains a set of element tags and values that are generated by default for the connection, schema, query, and release managers. This file represents the Repository Object Model (ROM) layer that contains information about the underlying repository structure.
- The `com.blazesoftware.repository.config.cfg` file contains the information for the base repository layer. It contains information on the services available for your repository. If you added a Version Manager and/or an Authorization Manager class to your configuration, the element tags and values for these classes would be displayed in this file.

If you create a private or local workspace for a versioned repository, the `com.blazesoftware.repository.config.cfg` file that is added to the root of the workspace directory contains repository connection information. In addition, the workspace user name is added to the configuration file.

If you selected the option to save your workspace or repository connection credentials, an element tag is added with a value of `true`.

If the repository connection you created is for a non-file BVS repository that was configured to allow the workspace and repository connection credentials to be different, an additional `<Impersonate/>` with the value of `true` is added within the versioning tags.

See:

- “Allowing Distinct Repository and Workspace Users” on page 49
- “Allowing Distinct Repository and Workspace Users When Using the Utility” on page 184

If you want to edit the repository files prior to creating the repository, see “[Editing a Repository Configuration in the Blaze Advisor IDE](#)” on page 39.

If you want to change the repository configuration after you have created your repository, see “[Changing an Existing Repository Configuration](#)” on page 144.

system Folder

A *system Folder* is also added to the root directory along with an `.attbs` file. The system Folder contains the query, filter, verification, and management properties templates and instances used to retrieve information about entries in your project, verify your rule project, and to filter project items from your project. See “[Managing Templates and Instances in Your system Folder](#)” on page 119.



Note If you are using a versioned repository with a private or local workspace, as soon as you connect to the workspace, a copy of the repository configuration files and the system Folder are added to your workspace directory. When you create a workspace using the `NdRomAdminUtil` utility, these files are automatically added to your workspace.

Admin Repository

You can also connect to the Admin Repository that contains the connection and configuration templates and instances for the repository types supported by Blaze Advisor. See ["Managing Repository Configuration and Connection Instances in the Admin Repository" on page 131](#).

Editing a Repository Configuration in the Blaze Advisor IDE

Blaze Advisor provides a set of pre-defined repository configurations for each default repository type it supports. Each pre-defined Blaze Advisor repository configuration contains the parameters for the persistence type. You can view the default Repository Types in the drop-down list on the first page of the Create a Repository wizard. If the repository configuration is for a persistence type with a versioning service, the details of the versioning services can be viewed in the Details page of the wizard.

You can edit the Details page when you want to add or delete a service for a pre-defined configuration. If you edit the information you see in the Details page, you are editing the configuration of the repository you are creating. These changes are stored in the repository configuration files automatically added to the root directory of your repository. These changes are not persisted to the underlying repository configuration instance that provides the default settings that you see in the Repository Type drop-down.

To make changes to the underlying configuration in the Admin Repository, see ["Editing Configurations and Connections Instances" on page 134](#).

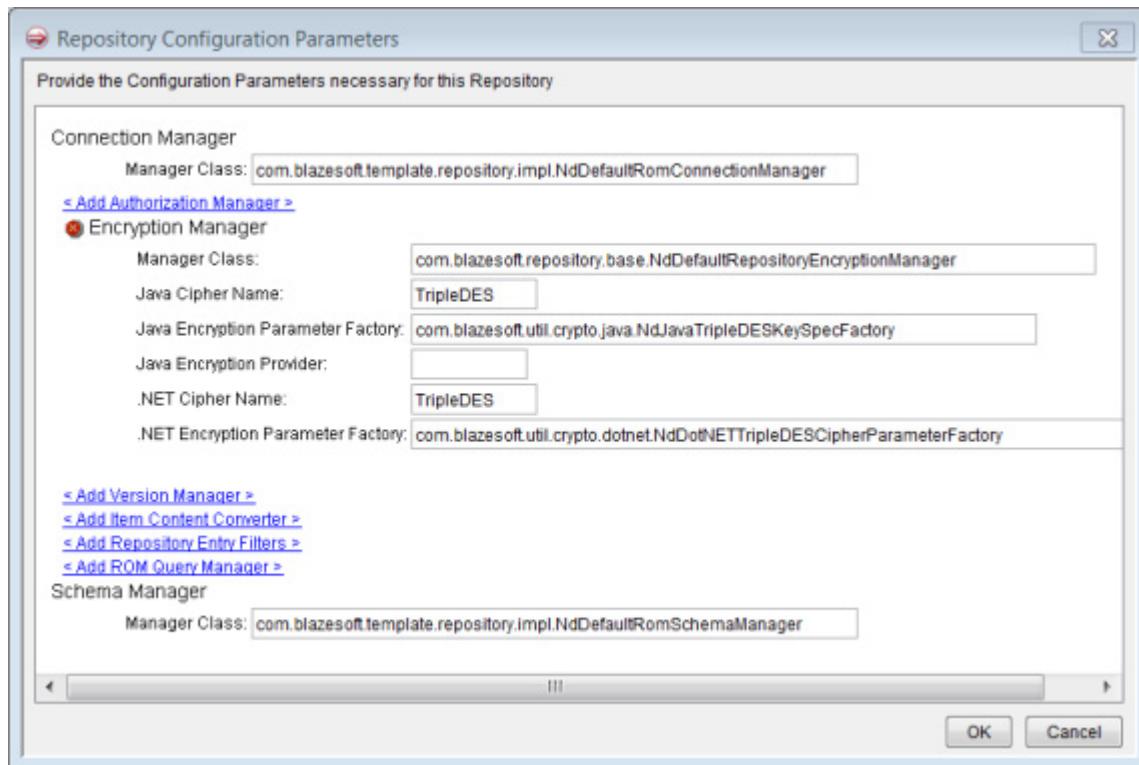
On the Details page, you can add an option by clicking a link. If you want to delete an optional parameter, click **Delete** icon . There are two options that you cannot delete:

- The connection manager class is used to manage the connection to the repository. The **Connection Manager** class field contains the default class name, `com.blazesoft.template.repository.impl.NdDefaultRomConnectionManager`. The connection manager makes it possible for the user to connect to a repository using the connection parameters or disconnect from a repository. The connection manager also supplies the client with a connection context that includes a schema manager and a query manager, if one is used.
- The schema manager is used to manage the type information of the repository contents. In the **Schema Manager** class field, retain the default class name, `com.blazesoft.template.repository.impl.NdDefault.RomSchemaManager`. See ["Schema Management of Type Information" on page 130](#).
- By default, the **Encryption Manager** is added to your repository to encrypt any passwords for your repository. See ["Password Encryption with the Encryption Manager" on page 44](#).
- By default, the **Item Content Compression Threshold KB** value is added to your repository. If you are using a MongoDB type repository, this compression threshold is used and can be edited, if you are using another repository type, this value is ignored. See ["Setting the Item Content Threshold" on page 326](#).

The changes you make in the Details page are displayed in one of two files generated in your repository directory. If you added a versioning or an authentication and authorization service to your repository, the `com.blazesoftware_repository_config.cfg` contains the configuration tags for the Version Manager or Authorization Manager. The `com.blazesoftware_rom_config.cfg` file contains the configuration tags and classes for the Connection Manager and the Schema Manager. If you added other optional configuration parameters such as a Query Manager, this class would also be listed in the `com.blazesoftware_rom_config.cfg` file. See “[Blaze Advisor Repository Contents and File Structure](#)” on page 37.

If you want to provide privileged users with the ability to set protections on repository entities, see “[Implementing Content Protection for Repository Items](#)” on page 413.

-  **Note** If you add a custom class to your configuration and realize that you have made an entry error after you have created your repository or you would like to add a configuration parameter after you have created your repository, you need to edit the `com.blazesoftware_rom_config.cfg` or the `com.blazesoftware_repository_config.cfg` file located in your repository directory. See “[Changing an Existing Repository Configuration](#)” on page 144.



To edit a repository configuration when creating a repository

- 1 Choose **Repository > New Repository**.
The Repository Type page is displayed.
- 2 Enter a name for the repository you want to create in the **Repository Nickname** field.

- 3 Choose an existing configuration from the **Repository Type** drop-down list.
- 4 Click **Details**.
 - a To add an optional configuration parameter click the link for the manager and either use the default class or enter the fully qualified name of the custom class in the field. To delete an optional parameter, click **Delete** icon . Parameters you cannot delete do not have a Delete icon next to them.



Important If you delete or alter any of the manager class names on the Details page, and immediately decide that you want to revert your changes, those changes cannot be reverted unless you exit the wizard and re-invoke it.

- b (Optional) Click **Add Authorization Manager** to add authorization services. Retain the default class name,
`com.blazesoftware.repository.base.NdRepositoryDefaultAuthorizationManager` or enter the fully qualified name of the authorization manager class you want to use. See "[Creating a Repository With a Custom Authorization Manager Class](#)" on page 402.
- c (Optional) Click **Add Version Manager** to add a versioning service and follow these steps:
 - d In the text field for the Version Manager class, select the class name for the type of versioning you want to use:
 - `com.blazesoftware.repository.generic.version.NdNativeRepositoryVersionManager`
Specifies the class used to implement Blaze versioning (BVS) for a File or Database repository and stores copies of each repository entry separately.
 - `com.blazesoftware.repository.generic.version.NdZipRepositoryVersionManager`
Specifies the class used to implement Blaze versioning (BVS) for a File or Database repository and stores copies in a single zipped archive.
 - `com.blazesoftware.repository.file.NdFileCVSWorkspaceVersionManager`
Specifies the class used to implement a CVS repository using a local workspace and connected to a CVS server.
 - `com.blazesoftware.repository.scm.wvcm.cc.NdCcVersionManager`
Specifies the class used to implement a ClearCase repository using a view and connected to a ClearCase server.
 - `com.blazesoftware.repository.scm.subversion.NdSvnWorkspaceVersionManager`
Specifies the class used to implement a Subversion repository using a workspace and connected to a Subversion server.

- e In the **Support Private Workspace** text field, retain `true` if you want to use private workspaces with a File or Database BVS repository.
Select `false` if you are creating a File BVS versioning with a shared workspace.
The Support Private Workspace option does not apply to the CVS repository or a Subversion or ClearCase repository, see “[Implementing a CVS Repository](#)” on page 235 or “[Implementing External SCM Repositories](#)” on page 253.
If you would like more information about implementing a repository with a BVS versioning service, see “[Implementing a Repository With a Blaze Versioning Service](#)” on page 209.
 - f (Optional) If you are using a Database BVS private repository, by default the **Allow distinct repository and workspace users** policy option that allows the workspace connection credentials to be different from your repository server connection credentials is selected. See “[Allowing Distinct Repository and Workspace Users](#)” on page 49.
You can clear the check box if you want your users to enter a common user name and password for their Blaze Advisor repository AND their Blaze Advisor workspace connections.
- 5 (Optional) You can select the **Save user and password** check box to use a link to quickly and easily connect to the workspace or repository in Blaze Advisor.
-  **Note** If you are using CVS versioning or Subversion or ClearCase versioning, the Support Private Workspace text field is ignored.
- a (Optional) Click **Add Item Content Converter** to enter the fully qualified name of an item content converter class.
The Item Content Converter can be used to convert content on read and write operations. For example, you can create a custom class and reference it in this field to compress the content prior to storage and decompress after retrieval, or encrypt the content before storage and decrypt after retrieval.
 - b (Optional) Click **Add Repository Entry Filter** to enter the fully qualified name of the custom repository entry filter class.
The Repository Entry Filter is used to filter any administrative files that a source control system has placed in a directory from displaying in the repository. For example, if you are using a CVS repository, you can create a custom class and reference it in this field so that the CVS folder does not appear in the Repository Explorer.
 - c (Optional) Click the **Add ROM Query Manager Class** field, retain the default class name:
`com.blazesoftware.template.repository.query.NdRomDefaultQueryManager` in the **Query Manager Class** field.
 - d Click **Add ROM Queries** to add an additional custom query class. Enter the fully qualified class name in the **Repository Query Factory Class** field. The class

name you enter in this field is enclosed within query factory class tags in the repository configuration file

You can add additional custom queries by clicking **Add ROM Queries**.

- 6 After you have completed making your changes, click one of the following items:
 - **OK** to return to the main wizard page.
 - **Cancel** to exit the wizard without creating a repository.

Saving User Name and Passwords

If you are creating a Blaze Advisor repository using one of the default types using versioning or authorization, the **Save user and password** check box option is displayed. This option saves the connection credentials for a workspace and/or repository server connection to your repository configuration file and to a connection instance file in the Admin Repository on your local machine. See [“Managing Repository Configuration and Connection Instances in the Admin Repository” on page 131](#).

- If this option is selected (default), you can automatically connect to your workspace or repository by clicking a repository connection name such as those listed in the Manage Connections wizard. You do not have to reenter your workspace credentials each time you need to connect to the workspace and/or repository server connection because each time you attempt to connect to the Blaze Advisor workspace or repository, the system checks the repository configuration file which stores the user name against the credentials stored in the connection instance file or the credentials you entered.
- If you do not select this option, your credentials are not stored locally and you are required to enter the connection credentials manually each time you want to connect.



Note By default, password encryption is enabled when user and password information is persisted in the Blaze Advisor IDE. See [“Password Encryption with the Encryption Manager” on page 44](#).

If you are creating a non-file BVS repository with private workspaces, and you may want to set a policy to allow the workspace credentials to be distinct from the repository server connection credentials, see [“Allowing Distinct Repository and Workspace Users” on page 49](#).

If you are creating an SCM repository and you select the Save workspace user and password option, the credentials are not persisted in the repository configuration but are saved to the connection instance in the Admin Repository. See [“Creating a View For Each User” on page 267](#).

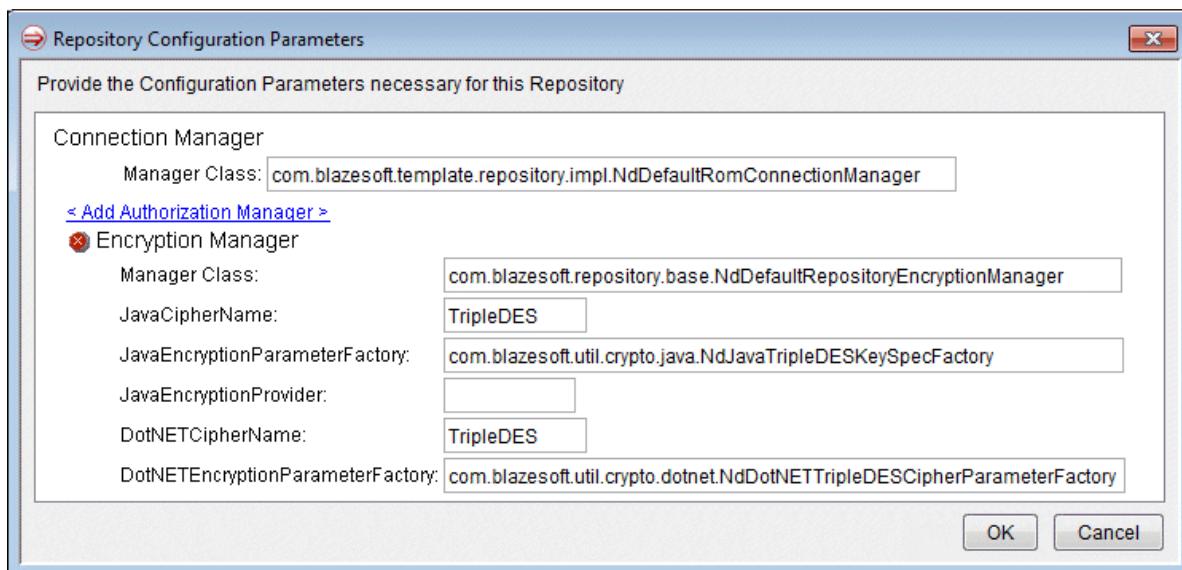
If you want to use the utility to create your repository and you want to save your workspace and/or repository credentials, see [“Saving User Names and Passwords Using the Utility” on page 183](#).

Password Encryption with the Encryption Manager

Password encryption assures that a password entered in a Blaze Advisor Wizard is not readable as plain text when the password is persisted to the associated configuration file, such as the connection credentials for a workspace, a repository configuration file, and the connection instance files in the Admin Repository. The user name and encrypted password are persisted when the **Save user and password check box** option is selected in the Create a Repository, New Repository Connection, Repository Connection Parameters, and New Workspace wizards.

Encryption Manager is enabled by default in the new repositories and workspaces you create. Encryption is supported for all types of repositories. Once Encryption Manager is enabled in a repository or workspace, the passwords you enter in all Blaze Advisor wizards will be encrypted by default. In addition to the wizards noted above, password encryption is supported in the wizards for a DatabaseBOM and Database Provider. When defining a remote server manager, the password entered in the System Definition **Deployment Manager** connection settings panel is encrypted in the server configuration (.server) and deployment manager (.dmanager) files. The password in an RMA connection file is encrypted as well.

The Encryption Manager is configured on the Repository Configuration Parameters page, that is available by clicking **Details** in the Create a Repository wizard. See "[Creating a Repository Using the Blaze Advisor IDE](#)" on page 28.



Encryption is supported for the Java and .NET platforms. A repository created or modified to use the default Encryption Manager configuration can be used with both Java and .NET clients. It is recommended that you do not modify the Encryption Manager configuration for the platform you are not using.

The default Encryption Manager class is `NdDefaultRepositoryEncryptionManager`. The name of the cipher and the cipher parameter factory class are specified separately for the two platforms. `TripleDES` is the default cipher for both Java and .NET. The cipher parameter factory classes provide additional configuration for the cipher on their

respective platforms and those classes are `NdJavaTripleDESKeySpecFactory` for Java and `NdDotNETTripleDESCipherParameterFactory` for .NET. Further detail on each field is supplied below.

Java Cipher Name is the name of a symmetric cipher implementation on the Java platform. The names of ciphers on the Java platform are defined by the Java Cryptography Extension, and are a standard part of the JDK.

Java Encryption Parameter Factory names an implementation of the interface `com.blazesoft.util.crypto.java.NdJavaEncryptionKeySpecFactory`. To use a Java cipher other than the default, TripleDES, you must provide an implementation of this interface that will work with the cipher.

For example, if you want to use `com.blazesoft.util.crypto.java.NdJavaPBEKeySpecFactory` as your Java Encryption Parameter Factory and the PBEWithMD5AndTripleDES as the Java Cipher Name, you must install the JCE unlimited strength jurisdiction policy files. If you do not download and install these policy files prior to using the PBEWithMD5AndTripleDES cipher, the default policy settings in the JRE will prevent its use.

Depending on your Java 7 or 8, you can download the policy files from the following sites:

- For Java 7

<http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html>

- For Java 8:

<http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html>

Java Encryption Provider is not shown in the configuration above, as its default value is null. This is specific to the Java implementation of the encryption framework, and allows you to specify a provider of ciphers in the Java Cryptography Architecture, as an alternative to using the default provider supplied with the JDK. (An article on implementing a provider in the Java Cryptography Architecture is available on the Oracle website.)

.NET Cipher Name is TripleDES which is the name of a symmetric cipher implementation on the .NET platform. This must be a subclass of the .NET class `System.Security.Cryptography.SymmetricAlgorithm`.

.NET Encryption Parameter Factor names an implementation of the interface `com.blazesoft.util.crypto.dotnet.NdDotNETEncryptionCipherParameterFactory`. To use a .NET cipher other than the default, TripleDES, you must provide an implementation of this interface that will work with that cipher.



Note This default implementation leverages a “core” cryptographic service which is adequate for most installations.

To disable the Encryption Manager, click the **Delete** icon next to **Encryption Manager** in the Repository Configuration Parameters page.

Adding Encryption Manager to an Existing Repository

Beginning with Blaze Advisor 7.1, Encryption Manager is installed by default for all new repositories. This section describes how to install Encryption Manager for repositories created with previous versions of Blaze Advisor. The repositories must first be upgraded, as described in "Migration to Blaze Advisor 7.x" in *Migration.pdf*.

The procedure for adding Encryption Manager to an existing expository follows the general procedure for adding and removing services from a repository in "[Changing an Existing Repository Configuration](#)" on page 144.

Two configuration files in <ADVISOR_HOME>/lib are edited. It is recommended that the original files are backed up prior to editing. The <RepositoryConfig> element in lib/romConfig.cfg is replaced with the <RepositoryConfig> element obtained from com.blazesoftware.repository.config.cfg in the repository folder (or the workspace folder, for versioned repositories) of the existing repository. Next, the <RepositoryEncryptionManagerConfig> element is copied into <RepositoryConfig> in lib/romConfig.cfg.

A standard file workspace connection file (lib/fileWorkspace.cfg) is edited as appropriate for connecting to an existing repository or workspace.

The NdRomAdminUtil -changeConfig option adds Encryption Manager to the existing repository or workspace. For revisioned repositories, the -commitConfig option commits changes to the repository configuration.

The process is straightforward and does not require specific knowledge of configuration file elements.

Preparing romConfig.cfg

After the editing of romConfig.cfg is completed, the <RepositoryConfig> element in romConfig.cfg will contain the <RepositoryEncryptionManagerConfig> element and the contents of the <RepositoryConfig> element (including child elements) copied from the com.blazesoftware.repository.config.cfg file that is located in the repository being updated.

To Prepare romConfig.cfg

- 1 Make a copy of <ADVISOR_HOME>/lib/romConfig.cfg to edit.
- 2 In romConfig.cfg, remove the existing <RepositoryConfig> element (this includes all child elements within <RepositoryConfig>).
- 3 Copy the entire <RepositoryConfig> element (including child elements within it) from <Repository_Folder>/com.blazesoftware.repository.config.cfg to romConfig.cfg. For versioned repositories, <Repository_Folder> is the workspace folder. (Note that steps 2 and 3 replace the content of the existing <RepositoryConfig> element in romConfig.cfg with the <RepositoryConfig> in com.blazesoftware.repository.config.cfg.)
- 4 Add the <RepositoryEncryptionManagerConfig> element shown below into the <RepositoryConfig> in lib/romConfig.cfg.

```

<RepositoryEncryptionManagerConfig>
  <EncryptionCipherFactory>
    <DotNETCipher> TripleDES </DotNETCipher>
    <DotNETCipherParameterFactory>
      com.blazesoftware.util.crypto.dotnet.NdDotNETTripleDESCipher
      ParameterFactory
    </DotNETCipherParameterFactory>
    <JavaCipher> TripleDES </JavaCipher>
    <JavaCipherParameterFactory>
      com.blazesoftware.util.crypto.java.NdJavaTripleDESKey
      SpecFactory
    </JavaCipherParameterFactory>
  </EncryptionCipherFactory>
  <RepositoryEncryptionManagerFactory>
    <JavaName> com.blazesoftware.repository.base.NdDefaultRepository
    EncryptionManager</JavaName>
  </RepositoryEncryptionManagerFactory>
</RepositoryEncryptionManagerConfig>

```

- 5 Save romConfig.cfg.

Editing the Workspace Connection File (fileWorkspace.cfg)

Configure fileWorkspace.cfg in the <ADVISOR_HOME>/lib folder.

To Configure the Workspace Connection File

- 1 Make a copy of the fileWorkspace.cfg to edit.
- 2 In fileWorkspace.cfg, edit the <Name>, <RepositoryFolder>, <RepositoryName>, <User>, and <Password> elements as appropriate. For versioned repositories <RepositoryFolder> is the Workspace folder. Add the <Password> element when the repository requires a password. Remove the <User> element when the user name is not required.

```

<RepositoryConnection>
  <Factory> com.blazesoftware.repository.file.NdFile
  RepositoryConnection </Factory>
  <Name> Sample Workspace Connection File </Name>
  <RepositoryFolder> C:\data\MyFileWorkspace </RepositoryFolder>
  <RepositoryName> MyFileWorkspace </RepositoryName>
  <User> user1 </User>
  <Password> pass1 </Password>
</RepositoryConnection>

```

- 3 Save fileWorkspace.cfg.

Using NdRomAdminUtil to Add Encryption Manager

The final step in adding Encryption Manager to an existing repository is to run NdRomAdminUtil.

To Add Encryption Manager to the Existing Repository

- 1 Open a command prompt window at <ADVISOR_HOME>/lib.
- 2 Run the command <ADVISOR_HOME>/bin/setenv.bat.
- 3 Run the following command at the prompt to add Encryption Manager to the repository or workspace.

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil changeConfig  
-workspaceConnection fileWorkspace.cfg -configuration romConfig.cfg
```
- 4 For versioned repositories, run the NdRomAdminUtil again, this time with the -commitConfig command option that is used to commit the workspace changes to the repository.

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil commitConfig  
-workspaceConnection fileWorkspace.cfg
```

If you have any other workspaces that connect to the repository, you need to run NdRomAdminUtil with the -updateConfig command so that the configuration files in those workspaces are updated.

Using NdRomAdminUtil to Encrypt a Password

The NdRomAdminUtil *encrypt* command is used to display the encrypted representation of the supplied password. The default cipher name and cipher parameter factory are described in ["Password Encryption with the Encryption Manager" on page 44](#).

Command, Argument, or Option	Description
encrypt	Command to encrypt a password. The encrypted password is written to command console.
-text	Argument takes the clear text password that is to be encrypted.
-cipherName	Option takes the name of the cipher. TripleDES is the default cipher for both Java and .NET. See "Password Encryption with the Encryption Manager" on page 44 .
-cipherParameterFactory	Option takes a class that implements the supported Java or .NET encryption parameter factory interface. See "Password Encryption with the Encryption Manager" on page 44 .
-encryptionManager	Option takes a class that implements an alternate encryption manager, such as com.example.MyEncryptionManager.
-workspaceConnection	Optional argument using the location of the connection configuration file for a workspace or repository configured with a Cipher Name, Cipher Parameter Factory, and Encryption Manager.
-verbose	Option prints out all detailed messages into the command console.

If you are creating a batch script with custom Java code that connects to a repository, the script must contain the password in encrypted format. Use the *encrypt* command to obtain the password in encrypted format.

The following examples show how you can use the encrypt command. When the command runs successfully, you should see the encrypted string displayed along with a success message.

To encrypt a given text string:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil encrypt -text  
<PlainText>
```

To encrypt a given string using a specified Cipher Name, Cipher Parameter Factory class and an Encryption Manager class:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil encrypt -text  
<PlainText> -cipherName TripleDES -cipherParameterFactory  
com.blazesoftware.util.crypto.java.NdJavaTripleDESKeySpecFactory -  
encryptionManager com.example.MyEncryptionManager
```

To encrypt a given text string using the Cipher Name, Cipher Parameter Factory class, and Encryption Manager configured in a workspace or repository connection:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil encrypt -text  
<PlainText> -workspace workspaceConnection.cfg -verbose
```

Allowing Distinct Repository and Workspace Users

When you create a non-file BVS repository with private workspaces in Blaze Advisor, you may want to set a policy that allows the workspace connection credentials and the repository server connection credentials to be different. If you set this policy, each of your users connect to the repository using their own workspace credentials and a set of common credentials for the repository connection. If you do not choose to use this policy, all your users are forced to use common credentials for both their workspace connection and their repository connection.

By default, when you click the Details button on the New Repository wizard, you see the Repository Configuration Parameters page where the Allow distinct repository and workspace users option is selected in the Version Manager section. If for some reason if this option is not displayed, select true from the Support Private Workspace drop-down list to display it. This option allows your users to have their own workspace connection credentials but use a common set of credentials for the repository connection. For example, if you are creating a Database BVS repository with private workspaces, a user can connect to the repository by using their own workspace connection credentials, "user A" and "pswd" and use the common database repository connection credentials, "scott" and "tiger" to connect to the repository server.

Alternatively, it is also possible to allow your users to connect to their workspace *and* the repository server using their workspace credentials. You need to inform your users about the connection policy you have set, before they create connections to the workspace and repository.

The policy to allow distinct repository and workspace users does not apply if you are using a non-versioned file, a file BVS shared repository, a file CVS repository or a Subversion or ClearCase repository.

If you want to use the `NdRomAdminUtil` utility to create a non-file BVS private repository using this policy, see "[Allowing Distinct Repository and Workspace Users When Using the Utility](#)" on page 184.

For information on connecting to a repository, see "[Connecting to a Blaze Advisor Repository From a Rule Maintenance Application](#)" on page 55.

Converting a Repository to Use Another Type

You can use commands in Blaze Advisor IDE or the `NdRomAdminUtil` utility commands `importWorkspace`, `exportWorkspace`, or `exportImportWorkspace` to convert a repository with or without versioning to another repository type with or without versioning. There are also utility commands to export a project to use another repository type. For specific information, see:

- "Importing a Blaze Advisor Contents from Another Blaze Advisor Workspace" in *DevelopingRuleProjects.pdf*
- "Exporting a Blaze Advisor Contents to Another Blaze Advisor Workspace" in *DevelopingRuleProjects.pdf*
- "Exporting and Importing a Workspace Using the Utility" in *DevelopingRuleProjects.pdf*
- "[Converting a Repository to and from a Database Repository](#)" on page 307
- "[Converting a Repository to and from a File CVS Repository](#)" on page 247
- "[Converting to a Cloud Compatible MongoDB Repository](#)" on page 381

CHAPTER 3

Connecting to a Blaze Advisor Repository

You use the FICO® Blaze Advisor® decision rules management system Manage Connections wizard in the IDE that contains a list of available workspace or repository connections to manage your workspace and repository connections. You can also use this wizard to create a new repository connection or modify, or delete any existing connections. When you are performing tasks such as importing or exporting projects and project contents or importing or exporting connections, you see this list of connections in other wizards.

This section contains the following topics:

- ["Workspace and Repository Connections" on page 51](#)
- ["Connecting to a Blaze Advisor Workspace or Repository" on page 52](#)
- ["Creating, Modifying or Deleting Blaze Advisor Repository Connections" on page 57](#)
- ["Importing a Blaze Advisor Workspace or Repository Connection" on page 61](#)
- ["Exporting a Blaze Advisor Workspace or Repository Connection" on page 62](#)

Workspace and Repository Connections

In Blaze Advisor a connection wizard contains a list of available workspace or repository connections. You can also use this wizard to create a new repository connection or modify, or delete an existing connection. A similar list of connections and commands are also available when creating workspaces, projects, or repositories using the New wizard or when you import or export a project or workspace contents, or workspace or repository connections using the Import, and Export wizards in Blaze Advisor. If you want to see the contents of a workspace or repository and perform some tasks at the repository level, you can use the Repository Explorer.

You indirectly create a new workspace or repository connection under the following circumstances:

- While creating a new repository.
- While creating a new local or private workspace.

You need to explicitly create a new workspace or repository connection under the following circumstances:

- If another user has created a non-versioned repository and you want to access its contents.

- If you have created a repository using the `NdRomAdminUtil` command-line utility.
After you have created a repository of any type using the command-line utility, you need to explicitly create a workspace or repository connection.
- If you have a connection file (.acn) to import for a workspace or repository connection
These connection files are not the same as the connection configuration files (.cfg) you use when you are creating a repository or workspace using the `NdRomAdminUtil` utility. See "[Writing Connection Configuration Files](#)" on page 169.

For information on	See
Logging onto an RMA and connecting to a workspace or repository.	"Connecting to a Blaze Advisor Repository From a Rule Maintenance Application" on page 55.
Specifying a workspace or repository connection in a deployed rule service.	"Connecting to a Blaze Advisor Repository From a Deployed Rule Service" on page 56.
Creating connection files for other users.	"Exporting a Blaze Advisor Workspace or Repository Connection" on page 62.

Connecting to a Blaze Advisor Workspace or Repository

There are several ways to connect to a Blaze Advisor workspace or repository to access its folders, projects, and project items:

- "Importing a Blaze Advisor Project Using a Wizard" in *DevelopingRuleProjects.pdf*
- "Importing a Blaze Advisor Project from the Repository Explorer" in *DevelopingRuleProjects.pdf*
- "[Connecting to an Existing Repository](#)" on page 52
- "[Connecting to a Blaze Advisor Repository From a Rule Maintenance Application](#)" on page 55
- "[Connecting to a Blaze Advisor Repository From a Deployed Rule Service](#)" on page 56

Connecting to an Existing Repository

There are several ways to connect to an existing workspace or repository in the Blaze Advisor IDE. The most direct way to selecting a connection from a list in the Repository Explorer. If you use this method, you see a tree-view of the entire workspace or repository contents. You use the Repository Explorer to open projects in the Project Explorer or add folders or projects to the overall structure of the workspace or repository. See "[Importing a Blaze Advisor Project from the Repository Explorer](#)" in *DevelopingRuleProjects.pdf*.

You also choose an existing workspace or repository connection from a list in the Blaze Advisor IDE when you create a project, import a project from another workspace or repository, import a workspace, export a project from another workspace or repository, export a repository connection, or export a workspace.

If you do not see a connection for a workspace or repository you want to use when attempting any of the tasks mentioned in the previous paragraph, you need to create a new connection for the workspace or repository in the Blaze Advisor IDE, see “[Creating a New Blaze Advisor Workspace or Repository Connection](#)” on page 58.

If you have a Comparison Query open in an editor, you can select an existing repository connection or create a new connection, see “[Selecting a Repository or Workspace Connection](#)” on page 53.

If you are connecting to a Database repository, you need to have the database driver in your classpath in the Blaze Advisor IDE, see “[Setting a CLASSPATH Using Blaze Advisor](#)” on page 286.



See Also

- “Importing a Blaze Advisor Project Using a Wizard” in *DevelopingRuleProjects.pdf*
- “[Creating, Modifying or Deleting Blaze Advisor Repository Connections](#)” on page 57
- “[Importing a Blaze Advisor Workspace or Repository Connection](#)” on page 61
- “[Exporting a Blaze Advisor Workspace or Repository Connection](#)” on page 62

Selecting a Repository or Workspace Connection

You can select an existing repository or workspace connection or create a new repository or workspace connection.

If you want to do any of the following task related to repository or workspace connections:

- Import a repository or workspace connection using a .acn file, see “[Importing a Blaze Advisor Workspace or Repository Connection](#)” on page 61.
- Modify a repository or workspace connection, see “[Modifying a Blaze Advisor Workspace or Repository Connection](#)” on page 59.
- Delete a repository or workspace connection, see “[Deleting a Blaze Advisor Repository or Workspace Connection](#)” on page 60.

To select an existing or create a new connection

- 1 If you want to select an existing repository or workspace connection, select the name of the connection from the Repository Connection list and click OK and return to the editor.

Alternatively, if you want to create a connection to a repository or workspace connection, click **New**.

The New Connection dialog opens.

- 2 Select **Existing workspace**, if you want to create a new connection to an existing workspace. If you want to create a new workspace and connection, go to Step 3.

The New Repository Connection wizard opens.

- a Enter a **Connection Name**.

This name will display in the Manage Connections wizard.

- b Select a workspace or repository type.

Workspaces and repositories can be File or Database. By default, workspaces for versioned repositories are of File type. If you are using a non-versioned repository, your repository is also your workspace.

If you choose to create a Database type connection, the connection is to a non-versioned database repository that exists in a table on a database server.

- c Click **Next**.

- d Enter the parameters for the workspace type you selected.

- e (Optional) If you are connecting to a workspace for a repository with BVS, CVS, ClearCase, or SVN versioning or an authorization manager, you will see some additional fields:

- Enter a user name and password.
- (Optional) Click Test Connection to verify if entered your user name and password correctly.
- (Optional) You can select the **Save user and password** check box to use a link to quickly and easily connect to the workspace or repository in Blaze Advisor. See "[Saving User Name and Passwords](#)" on page 43.

- f Click **Finish**.

You see a message confirming that the connection has been created.

- g Click **OK**.

- 3 Select **New workspace**, if you want to create a new workspace and connection.

The New Workspace wizard opens.

- a Select the repository type for the workspace you want to create.

- b Enter a connection name for your workspace.

After the workspace is created, the connection name you entered here is displayed in the connection lists.

- c Click **Next**.

- d On the Connection Parameters page, enter the connection settings for the repository and the location information of the workspace.

- e (Optional) You can select the **Save user and password** check box to use a link to quickly and easily connect to the workspace or repository in Blaze Advisor. See "[Saving User Name and Passwords](#)" on page 43.
 - f Click **Finish**.
- 4 Select the new connection you added to the Repository Connection list and click **OK** and return to the editor.

Connecting to a Blaze Advisor Repository From a Rule Maintenance Application

The Sign-in page is included in the default set of RMA generation options so that you can use the information in your repository configuration to control who can use the RMA. If you do not include a Sign-in page when you generated the RMA, when users run the server and open an RMA in a browser by entering a URL or through the IDE, they are immediately connected to the repository and the RMA files.

If a Sign-in page is used and users successfully log onto an RMA, they can begin to work on projects that they can access from the RMA Home page. Users can successfully log on if their user name and password can be authenticated either through LCM Access Control if there is no versioning system configured with the repository or through a versioning system or through a combination of the LCM Access Control and a versioning system. See "Logging onto an RMA with Life Cycle Management" in *DevelopingRuleMaintenanceApplications.pdf*.

On an application-level here when users successfully log on, they connect to the workspace or repository. If users are connecting to a repository with no versioning or a File Repository (BVS versioning - shared) or a File Repository (with Authorization manager), they directly connect to the repository via the Sign-in page. If users are connecting to a repository with versioning, they connect to the repository via a workspace that is created in most cases, specifically for their work in the RMA.

A new workspace folder is not created, if the Connect to current workspace for this user option is selected in the Workspace tab when the RMA is generated. This option only applies to the user who is generating the RMA. See "Selecting Workspace Options" in *DevelopingRuleMaintenanceApplications.pdf*.

A new workspace folder is also not created for users who are signing into an RMA connected a ClearCase workspace. Setting up the workspace connections for a ClearCase repository requires additional tasks to be completed. See "[Creating RMA ClearCase Workspace Connections](#)" on page 277.

When you generate an RMA using a project stored in a versioned repository, depending on the repository type, you can specify the location where those RMA workspaces are created. If you are generating an RMA using a project stored in one of the following versioned repository types:

- File Repository (BVS versioning - private)
- File Repository (CVS versioning)
- File Repository (Subversion versioning)

- File Repository (ClearCase versioning*)
- Database Repository (BVS versioning - private)
- MongoDB Repository (BVS versioning - private local file workspace)

Because the workspaces are of file type, in the RMA Generator wizard, you use the Workspaces directory field to specify a location on a file system where you want the workspaces to be created. For example, C:/Workspaces. When users log onto the RMA, a workspace is created in this location. The workspace directory name is the user name entered. For example, C:/Workspaces/userA. A copy of the workspace files are added to this directory.

*If you want to generate an RMA from a project in a repository connected to an IBM Rational ClearCase server, you can choose between two methods for creating connections so that your users can connect to their workspaces in an RMA, see ["Generating an RMA with a ClearCase Workspaces Directory Parameter" on page 277](#).

If the repository connection credentials are not the same as the workspace credentials because the Allow Distinct Repository and Workspace Users policy was set in the repository configuration. For authentication purposes you are prompted to enter the additional repository credentials. See ["Allowing Distinct Repository and Workspace Users" on page 49](#).

If the RMA is generated from a project stored in a MongoDB (BVS versioning - private) where the workspaces are not of file type but MongoDB type, when users log onto the RMA, an RMA workspace is created for them in the MongoDB instance according to the Workspace Base Name specified. See ["RMA Generation for MongoDB Repositories and Workspaces" on page 385](#).

When you move your RMA files to an application server, you need to ensure that you include all the jar files necessary for your RMA components, see ["Moving Files to Your Application Server" in *DevelopingRuleMaintenanceApplications.pdf*](#).

Connecting to a Blaze Advisor Repository From a Deployed Rule Service

If you deploy a rule service from a project in most repositories including File BVS repositories with a shared workspace, File, MongoDB, or Database BVS repositories with private workspaces, File CVS repositories, File Subversion or File ClearCase repositories with local workspaces, and non-versioned repositories, the connection information is included in the server configuration (.server) file that was generated when you used the Quick Deployer wizard.



Important You need to consider how the network connection and the repository type you choose for your project will impact performance, see ["Network Connection and Repository Type Considerations For Rule Service Deployment Performance" on page 57](#).

The server configuration file contains information about the repository location, the repository connection class, project name, and the user name and password used to connect to the repository in nested within <WorkspaceConnection/> element tags of the <RulesProjectInnovatorLoaderFactory/> tags.

If you are using non-file BVS repository with private workspaces, the workspace connection credentials are included in the nested <Credentials/> element tags within <WorkspaceConnection/> nested within the <RulesProjectInnovatorLoaderFactory/> tags in the .server file.

If you are using a deployment manager, by default, when you deploy rule service, the information about the workspace connection is stored in the deployment manager file (.dmanager) file nested within <WorkspaceConnection/> element tags of the <RulesProjectInnovatorLoaderFactory/> tags. Similarly, if you are using non-file BVS repository with private workspaces, and use a deployment manager, by default, the workspace connection credentials are included in the nested <Credentials/> element tags within <WorkspaceConnection/> nested within the <RulesProjectInnovatorLoaderFactory/> tags in the .dmanager file.

Network Connection and Repository Type Considerations For Rule Service Deployment Performance

For best results when deploying a rule service, you need to use a low latency high speed network connection.

To obtain the best performance for a given network resource, we recommend the following repository types (in descending order from best performance to poorer performance):

- MongoDB Repository (BVS versioning - shared) or MongoDB Repository (BVS versioning - private) where the workspaces are stored in the MongoDB database.
- Database Repository (non- versioned).
- File Repository (BVS private, CVS, SVN, or ClearCase) where the File workspace on a network share.

Creating, Modifying or Deleting Blaze Advisor Repository Connections

If you need to create or maintain Blaze Advisor workspace and repository connections for your group, you can use the Manage Connections wizard for most tasks. Your users can create connections to the Blaze Advisor workspaces or repositories by using connection information you supply or by importing a connection file for the workspace or repository you want them to use.



Note You cannot simultaneously connect to the same Blaze Advisor workspace or repository in the same Eclipse workspace.

- “[Creating a New Blaze Advisor Workspace or Repository Connection](#)” on page 58
- “[Modifying a Blaze Advisor Workspace or Repository Connection](#)” on page 59
- “[Deleting a Blaze Advisor Repository or Workspace Connection](#)” on page 60
- “[Importing a Blaze Advisor Workspace or Repository Connection](#)” on page 61
- “[Exporting a Blaze Advisor Workspace or Repository Connection](#)” on page 62

If you used the `NdRomAdminUtil` utility to create a repository, you need to create a workspace or repository connection in Blaze Advisor. See “[Creating a New Blaze Advisor Workspace or Repository Connection](#)” on page 58.

Creating a New Blaze Advisor Workspace or Repository Connection

You can create a new connection to an existing Blaze Advisor workspace or repository created using the Blaze Advisor IDE to gain access to the workspace or repository contents. You cannot use the `NdRomAdminUtil` utility to create workspace or repository connections. If you attempt to do this, you will see an error message.

Although Blaze Advisor supports creating multiple connections to the same non-versioned File repository, you cannot simultaneously use more than one connection to the same non-versioned Blaze Advisor workspace in the same Eclipse workspace. For example, you cannot have imported project from both ConnectionA and ConnectionB in the same Eclipse workspace at the same time but you can delete the imported projects from ConnectionA and then import projects from ConnectionB. You can also use different connections to the same Blaze Advisor non-versioned workspace by using each connection in a different Eclipse workspace.

When you use the connection to a local or private workspace, in most cases the workspace type is file. If you are creating a connection to a Database repository or workspace, you need to have the appropriate driver class in your global classpath. See “[Setting a CLASSPATH Using Blaze Advisor](#)” on page 286.

To create a new workspace or repository connection

- 1 Select **Repository > New Repository Connection** or **Repository > Manage Connections** and click **New**.
Alternatively, from either the **File** menu or the Repository Explorer, select **New > Other** and expand the **Blaze Advisor** folder to select **New Repository Connection**.
- 2 Enter a **Connection Name**.
This name will display in the Manage Connections wizard.
- 3 Select a workspace or repository type.
Workspaces and repositories can be File, or Database, or MongoDB. The default is File.
See “[Determining the Workspace Type for Your Connection](#)” on page 59.
- 4 Click **Next**.
- 5 Enter the parameters for the workspace type you selected.

- 6 (Optional) If you are connecting to a workspace for a repository with BVS, or CVS, ClearCase, or SVN versioning or an authorization manager, or is a MongoDB repository type, you will see some additional fields:
 - a Enter a user name and password.
 - b (Optional) Click Test Connection to verify if entered your user name and password correctly.
 - c (Optional) You can select the **Save user and password** check box to use a link to quickly and easily connect to the workspace or repository in Blaze Advisor. See "[Saving User Name and Passwords](#)" on page 43.
- 7 Click **Finish**.
You see a message confirming that the connection has been created.
- 8 Click **OK**.
If you are creating connections for others to use, you need to export the connection as a file. See "[Exporting a Blaze Advisor Workspace or Repository Connection](#)" on page 62.

Determining the Workspace Type for Your Connection

When you are creating a new connection to an existing repository or workspace, you select a Workspace Type that is the same as the workspace or repository you want to access.

- If you are using a non-versioned repository, your workspace is the same as your repository and therefore is the same type.
- Workspaces for most versioned repositories including Database repositories are of File type.
- If you are using a MongoDB repository that is non-versioned, or versioned using a private or shared workspace stored in MongoDB, your workspace type or repository type is MongoDB.
- If you are using a MongoDB repository that is versioned using a private local file workspace stored on a file system, your workspace type is File.
- If you choose to create a Database type connection, the connection is to a non-versioned database repository that exists in a table on a database server.

Modifying a Blaze Advisor Workspace or Repository Connection

You can modify an existing workspace or repository connection using the Manage Connections wizard. For example, if you rename or move the workspace or repository directory, you must modify any existing workspace or repository connections in order to use it. Before you modify your connection, you need to ensure that any projects you have open do not use the workspace or repository connection you are modifying. If the projects use the workspace or repository connection you are modifying, you need to delete project from the Eclipse workspace and reimport them after you have made the

modifications to the connection otherwise you may find that you are unintentionally making changes to project files stored in an workspace or repository location where you no longer have access.

To modify an existing workspace or repository connection

- 1 Select **Repository > Manage Connections**.
- 2 Select the workspace or repository connection you want to modify and click **Modify**.
- 3 If you have any projects belonging to the connection you want to modify, you see a Modify Connection Warning dialog window that lists the projects associated with the connection.

You see this warning because modifying a workspace or repository connection may affect the ability to open these projects.

Select one of these options:

- **Close and reopen projects that use this connection.**
- **Remove all projects that use this connection from the Eclipse workspace.**
- **Cancel modification of this connection.**

- 4 (Optional) If you are connecting to a workspace for a repository with BVS, CVS, ClearCase, or SVN versioning or an authorization manager, you will see some additional fields:
 - a Enter a user name and password.
 - b (Optional) Click **Test Connection** to verify if entered your user name and password correctly.
 - c (Optional) You can select the **Save user and password** check box to use a link to quickly and easily connect to the workspace or repository in Blaze Advisor. See ["Saving User Name and Passwords" on page 43](#).
For the MongoDB Repository Settings page, this check box is known as **Save database user and password**.



Important If you remove or modify an existing repository or workspace connection, you cannot undo your action but you can create a new connection to the previous repository or workspace location assuming the workspace or repository still exists.

Deleting a Blaze Advisor Repository or Workspace Connection

You can remove an existing workspace or repository connection using the Manage Connections wizard. If you delete a repository or workspace connection, it may make the projects in the workspace or repository unaccessible until a new connection is created.

To delete a repository or workspace connection

- 1 Select **Repository > Manage Connections**.
- 2 Select the connection you want to delete and click **Delete**.
If you have any projects belonging to the connection you want to delete, you see a Delete Connection Warning dialog window that lists the projects associated with the connection.
- 3 Select one of these options:
 - **Close all open projects that use this connection.**
 - **Remove all projects that use this connection from the Eclipse workspace.**
 - **Cancel deletion of this connection.**



Important If you remove an existing repository or workspace connection, you cannot undo your action but you can create a new connection to the repository or workspace.

Importing a Blaze Advisor Workspace or Repository Connection

You can import a connection file (.acn) to connect a workspace or repository. After you import a workspace or repository connection, the connection is listed on any wizard pages where you need to select a source or target workspace or repository to complete a task. A connection file is an XML file that contains the parameters and values for the workspace or repository connection. You can export a workspace or repository connection as a file. See ["Exporting a Blaze Advisor Workspace or Repository Connection" on page 62](#).



Important The connection (.acn) file should not be confused with the connection configuration file (.cfg) that you use when you want to create a repository or workspace using the NdRomAdminUtil utility.

To import a workspace or repository connection

- 1 Select **File > Import** and expand the **Blaze Advisor** folder.
- 2 Select **Repository Connection**.
- 3 Click **Next**.
- 4 Enter or browse to the location of the .acn file you want to import.
- 5 Enter a name for the connection in the **Import As** field.

The name you enter here is added to the list of available connections in the Manage Connections dialog window.

6 Click **Finish.**

You can use this connection to when you use a wizard to create a new project, new workspace, or import or export project or workspace contents.

Exporting a Blaze Advisor Workspace or Repository Connection

You can export a connection file (.acn) is an XML file that contains connection parameters and values for a workspace or repository. In general, you export a connection file so that you can store the connection information elsewhere on your system or to distribute it to other users. See ["Editing a Generated Connection File" on page 63](#).

If you create connection files for other users, they connect to the workspace or repository by importing the file you provide. See ["Modifying a Blaze Advisor Workspace or Repository Connection" on page 59](#).

To export a connection

- 1 Select **File > Export** and expand the **Blaze Advisor** folder and select **Repository Connection**.
- 2 On the **Repository Connections** page, you can either select a connection from your connection list to export or import an existing connection to add to your list of connections:
 - Select the connection file to export and click **Next**.
 - **Import** a connection to add to your list of connections for export.
If you click import, a File selection window opens where you can browser to the location of your .acn file.

- 3 On the **Connection Parameters** page, select and edit the parameters for the connection.

The type of the repository connection you selected from the list determines what you see on the Connection Parameters page.

You may wish to delete the user and password from the connection parameters before you export the connection so that your users can provide their own, or you may want to edit these fields for each user. You can also edit the connection file after you export it. See ["Editing a Generated Connection File" on page 63](#).

If you are exporting a connection to a Database BVS repository with private workspaces, the values in the User name and Password fields are used to connect to the repository server while the Local Folder field references a workspace location.

- 4 (Optional) Click **Advanced** if you want to see the complete list of connection parameters and values for the repository configuration. These parameters and values are included in the connection file (.acn) you export.

If the option to save your user name and password is selected, the existing repository connection credentials are used each time you connect to the workspace or repository.

- 5 Click **Next**.

A **Select the Location of the Exported Connection** window is displayed.

- a Navigate to the directory location where you want to save your exported connection file.
- b Enter a name for the connection file and click **Save**.



Tip Whenever possible try to use the repository connection name for the connection file. This will make it easier for your users to remember which repository connection they need to select after they import the connection.

When you look in the directory location where you exported your file, you should see the connection file name you specified appended with an .acn extension.

You can send this file to another user or place it in a shared network directory so that the connection file can be imported and used to connect to the repository. You can also edit the contents of this file. See "[Editing a Generated Connection File](#)" on page 63.

- 6 Click one of the following items:
 - **Finish** to exit the wizard.
 - **Cancel** to exit the wizard without saving any of your changes.
 - **Back** to return to the previous page.
- 7 Repeat Steps 1 to 6 for each user who needs access to the repository and export the file.

Editing a Generated Connection File

You may wish to edit the exported connection file (.acn) to provide each user with a workspace or repository connection tailored to their requirements. For example, you can edit the connection file to provide a confidential user name and password for each user before distributing the file. If you are using a non-file BVS repository that has been configured to allow distinct repository and workspace users, the workspace credentials do not need to be the same as the repository credentials. See "[Allowing Distinct Repository and Workspace Users](#)" on page 49.

To edit a generated connection file

- 1 Open the exported connection file in a text editor.
- 2 Search the connection file using the term, "username".
If you are using a non-versioned repository and have found the following tags,

```
<instance ref='com.blazesoftware.repository.username'>
<value>user</value>
</instance>
```

Edit the text string for the <value> element with the user name that you assigned this user.
If you are using a BVS repository with private workspaces, and have found the following tags,

```
<instance ref='com.blazesoftware.repository.workspaceuser'>
```

Edit the text string for the <value> element with the user name that you assigned this user.
- 3 (Optional) If you are using a password with your repository connection, search the connection file using "password".
When you have found the following tags,

```
<instance ref='com.blazesoftware.repository.password'>
<value>password</value>
</instance>
```

Edit the text string for the <value> element with the password that you assigned this user.
Alternatively, if you are using a BVS repository with private workspaces, and have found the following tags,

```
<instance ref='com.blazesoftware.repository.workspacepassword'>
```

Edit the text string for the <value> element with the user name that you assigned this user.
- 4 (Optional) If you are using a versioned repository with a private or local workspace, you can edit the value in the following tags,

```
<instance ref='com.blazesoftware.repository.workspacefolder'>
<value><absolute path to the workspace folder></value>
</instance>
```
- 5 Save your changes.

CHAPTER 4

Administering Blaze Advisor Workspaces and Repositories

You can perform several tasks when administering a FICO® Blaze Advisor® decision rules management system repository. The topics in this include:

- “Importing or Exporting Projects and Workspace Contents Using the Utility” on page 65
- “Releasing a Project Using the Utility” on page 76
- “Publishing a Project Using the Utility” on page 79
- “Managing Versioned Project Content Using the Utility” on page 83
- “Managing Files Using the Utility” on page 95
- “Managing Templates and Instances in Your system Folder” on page 119
- “Managing Repository Configuration and Connection Instances in the Admin Repository” on page 131
- “Synchronizing the system Folder” on page 137
- “Changing an Existing Repository Configuration” on page 144
- “Loading Custom Classes” on page 152
- “Embedding a Digital Signature Into a Configuration” on page 157

You can also perform the following tasks:

- “Creating a Blaze Advisor Repository Using the Utility” on page 33
- “Creating a Local or Private Workspace” on page 226
- “Converting a Repository to Use Another Type” on page 50

If you want to learn about that tasks for creating and managing project contents, see “Creating and Managing Project Content” in *DevelopingRuleProjects.pdf*.

Importing or Exporting Projects and Workspace Contents Using the Utility

You can import or export Blaze Advisor projects or workplace or repository contents using an existing workspace or repository connection by using either the Blaze Advisor IDE or a utility. The majority of the topics in this section use the

`NdRomAdminUtil` utility. However, in tasks where the `ImporterExporter` utility is used, it is noted by name. In most cases you can also use the similar commands in Blaze Advisor IDE to complete these tasks in this section. For your convenience where applicable the links are included in the following topics:

- “Importing a Project from an .adv File Using the ImporterExporter Utility” on page 66
- “Importing a Project from a Zip File Using the Utility” on page 68
- “Exporting a Project as an .adv File Using the ImporterExporter Utility” on page 69
- “Exporting a Project to a Zip File Using the Utility” on page 71
- “Exporting and Importing a Project” on page 72
- “Importing a Workspace from a Zip File Using the Utility” on page 74
- “Exporting a Workspace to a Zip File Using the Utility” on page 74
- “Exporting and Importing a Workspace Using the Utility” on page 75

Importing a Project from an .adv File Using the ImporterExporter Utility

You can use the `com.blazesoftware.template.repository.util.` utility in a command prompt to import Blaze Advisor project that was exported as a .adv file. You can also use the Import wizard in the Blaze Advisor IDE, see “Importing a Blaze Advisor Project from an .adv File” in *DevelopingRuleProjects.pdf*.

When you use the `NdImporterExportUtil` utility, the required arguments are based on the type of repository you want to use in the current version of Blaze Advisor.

Import Utility Arguments

The `NdImporterExporterUtil` utility requires arguments which must be specified using this format: <argument>=<value>. Each argument must be separated by a space. This table describes the arguments:

Argument	Description	Example
-import <pathToAdvisorProjectFile>	Specify the location of the .adv file you want to import.	-import "C:/Blaze/ Advisor/testProj.adv"
-project <repositoryFullPath> or -repositoryConnection <connectionParameters> See " Connection Parameter Sub-arguments " on page 67. Note The -repositoryConnection is the argument used when you want to specify repository connection parameters and values for a repository type.	The path or connection parameters to the file repository where you want your project to be imported. These two arguments are mutually exclusive. See the table that follows for the list of arguments for specific repository types.	-project c:/Blaze/Advisor/ testRepository - or - -repositoryConnection repositoryFolder="C:/ Blaze/Advisor/myRepo" user=user1 password=psswd -projectDirectory "/ myProject" - projectName importedProj

Connection Parameter Sub-arguments

This table shows the arguments for each type of repository that can be used with the -repositoryConnection argument of the `NdImporterExporterUtil` utility. Each argument needs to be followed by a = sign, the argument, and then a space, such as:
 repositoryFolder = c:/Blaze/Advisor/examples/ruleMaintenance/repository
 projectDirectory=/projectForClientA projectName=projectA

Repository type	Allowed arguments
File	repositoryFolder, user, password Note: repositoryFolder path can be absolute or relative.
Database	accessor, repositoryName, url, user, password, driver, table



Note If the repository you want to import your project into is a CVS, ClearCase, SVN or File, Database or MongoDB BVS repository with a local or private file workspace, the value for the `repositoryName` or `repositoryFolder` is the path to the workspace, and not the path to the repository.



Tip If your Advisor project contains an imported Java class, you can set the classpath in the Classpath panel before migrating the project into the current release. Alternatively, you can set the classpath in the project once it is migrated into the current release.

To import a project from an .adv file using the ImporterExporter utility

- 1 Create a repository in the current release of Blaze Advisor.
(The `NdImporterExporterUtil` utility does not create one for you.)
- 2 Open a command prompt window and cd to <ADVISOR_HOME>\bin.
- 3 Enter this text: `setenv`

- 4 Press **Enter** on your keyboard.
- 5 Enter the name of the utility with the arguments you wish to use. For example:
`java com.blazesoftware.template.repository.util.NdImporterExporterUtil -import C:/Blaze/Advisor/examples/rules/base/function1/function1.adv -repositoryConnection repositoryFolder=C:/Blaze/Advisor/lib/NewRepository -projectDirectory /MyFolder - projectName function1`
If the import operation is successful, you will see a confirmation message in the command prompt window.
- 6 Open the Repository Explorer to select a project. See “Importing a Blaze Advisor Project from the Repository Explorer” in *DevelopingRuleProjects.pdf*.
Alternatively, you can use the Import wizard to import a project in the Eclipse workspace. See “Importing a Blaze Advisor Project Using a Wizard” in *DevelopingRuleProjects.pdf*.

Importing a Project from a Zip File Using the Utility

You can import a project from a .zip file to a directory location in a workspace using the `NdRomAdminUtil` utility by using the command `importProject`. You can also use the Export wizard in the Blaze Advisor IDE, see “Importing a Blaze Advisor Project from a .zip File” in *DevelopingRuleProjects.pdf*.

When you use the utility, you need to reference the paths to the .zip file containing the project contents you want to import and the paths to the connection configuration file for target workspace. When the project folders and files are extracted from the .zip file, they are integrated with any existing folders and files in the workspace directory. If you are working with a local or private workspace, you need to check in the imported files to the repository.



Important If the workspace already contains a project with the same name, another project will be created. While the display name of the two projects will be identical, the project that is imported will have `_1` appended to the storage name.

Command, Argument, or Option	Description
<code>importProject</code>	Command for importing a project from a .zip file.
<code>-workspaceConnection</code>	A pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See “ Writing Connection Configuration Files ” on page 169.
<code>-zipFilePath</code>	Argument takes a pathname to the .zip file whose contents you want to import into the workspace.

Command, Argument, or Option	Description
-directoryLocation	Argument takes a directory location relative to the root directory in the workspace where you want to import your project files. Example: “/Project_A Folder”
-verbose	Option prints out all detailed messages into the command console.



Note Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, use double quotes.

Example:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil importProject
    -workspaceConnection "C:/Repo_Connections/FileWksp.cfg" -directoryLocation
    "/Project_1010" -zipFilePath "C:/Blaze/Advisor/ZipFiles/WkspContents.zip" -
    verbose
```

Exporting a Project as an .adv File Using the ImporterExporter Utility

You can use the

`com.blazesoft.template.repository.util.NdImporterExporterUtil` utility to export a project that was created in the current release of Blaze Advisor as an .adv file. You can also use the Export wizard in the Blaze Advisor IDE. See “Exporting a Blaze Advisor Project to an .adv File” in *DevelopingRuleProjects.pdf*.

When you use the `NdImporterExporterUtil` utility to export a project, the arguments you use are based on the repository type where the project is currently located. See “[Export Utility Arguments](#)” on page 69.

Export Utility Arguments

The `NdImporterExporterUtil` utility requires arguments which must be specified like this: <argument>=<value>. Each argument must be separated by a space. This table describes the arguments:

Argument	Description	Example
<p>-project <repositoryFullPath> or -repositoryConnection <connectionParameters></p> <p>See “Connection Parameter Sub-arguments” on page 70.</p> <p>NOTE: The -repositoryConnection is the argument used when you want to specify repository connection parameters and values for a repository type.</p>	<p>The path or connection parameters to the repository where the project you want to export is located.</p> <p>These two arguments are mutually exclusive. See the table that follows for the list of arguments for specific repository types.</p>	<pre>-project c:/Blaze/Advisor/testRepository - or - -repositoryConnection repositoryFolder="C:/Blaze/Advisor/myRepo" user=user1 password=psswd -projectDirectory "/myProjects" -projectName Proj1</pre>
-export <pathToAdvisorProjectFile>	Specify the location where you want the .adv file to be exported.	-export "C:/Blaze/Advisor/testProj.adv"

-  **Note** Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, use double quotes.

Connection Parameter Sub-arguments

This table shows the arguments for each type of repository that can be used with the -repositoryConnection argument of the `NdImporterExporterUtil` utility. Each argument needs to be followed by a = sign, the argument, and then a space, such as:
`repositoryFolder = c:/Blaze/Advisor/examples/ruleMaintenance/repository`
`projectDirectory=/projectForClientA` `projectName=projectA`

Repository type	Allowed arguments
File	repositoryFolder, user, password Note: repositoryFolder path can be absolute or relative.
Database	accessor, repositoryName, url, user, password, driver, table

-  **Note** If the repository you want to import your project into is a CVS, ClearCase, SVN or File, Database or MongoDB BVS repository with a local or private file workspace, the value for the `repositoryName` or `repositoryFolder` is the path to the workspace, and not the path to the repository.

-  **Tip** If your Advisor project contains an imported Java class, you can set the classpath in the Classpath panel before migrating the project into the current release. Alternatively, you can set the classpath in the project once it is migrated into the current release.

To export a project as an .adv file using the ImporterExporter utility

- 1 Create a repository in the current release of Blaze Advisor.
(The `NdImporterExporterUtil` utility does not create one for you.)
- 2 Open a command prompt window and cd to `<ADVISOR_HOME>\bin`.
- 3 Enter this text: `setenv`
- 4 Press **Enter** on your keyboard.
- 5 Enter the name of the utility with the arguments you wish to use. For example:

```
java com.blazesoftware.template.repository.util.NdImporterExporterUtil -  
repositoryConnection repositoryFolder=C:/Blaze/Advisor/lib/NewRepository  
-projectDirectory /MyFolder - projectName function1  
-export C:/Blaze/Advisor/examples/rules/base/function1/function1.adv
```

If the export operation is successful, you will see a confirmation message in the command prompt window.

You can now send the .adv file to another user to import the project into a different workspace or repository or store the file on your network.

Exporting a Project to a Zip File Using the Utility

You can export a project from a .zip file using the `NdRomAdminUtil` utility so that other users can import the file into their workspaces. To export a project from a .zip file, you use the command `exportProject` and you reference the path to the connection configuration file for the source workspace, the project location, and the pathname to the .zip file where want to export your project contents. You can also use the Blaze Advisor IDE, see “Exporting a Blaze Advisor Project to a .zip File” in *DevelopingRuleProjects.pdf*.

If you import projects into a private or local workspace, you need to check the files into the repository. When the folders and files of the .zip file are imported, they will be integrated with existing folders and files in the current workspace. See “Importing a Blaze Advisor Project from a .zip File” in *DevelopingRuleProjects.pdf*.

Command, Argument, or Option	Description
<code>exportProject</code>	Command for exporting a project to a .zip file.
<code>-workspaceConnection</code>	A pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See “ Writing Connection Configuration Files ” on page 169.
<code>-projectLocation</code>	Argument takes the project location relative to the root directory in the workspace. You need to specify the Project folder name and the project name in the path. Example “/Project_A Folder/Project_A”

Command, Argument, or Option	Description
-zipFilePath	Argument takes a pathname to the .zip file where you want to export the contents of your project.
-verbose	Option prints out all detailed messages into the command console.



Note Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, use double quotes.

Example:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil exportProject  
-workspaceConnection "C:/Repo_Connections/FileWksp.cfg" -projectLocation "/  
Project_A Folder/Project_A" -zipFilePath "C:/Blaze/Advisor/ZipFiles/  
ProjectContents.zip" -verbose
```

Exporting and Importing a Project

You can export and import contents from a project location in a source workspace to a directory location in a target workspace in one `NdRomAdminUtil` utility command. To export and import a project from a directory, you use the command `exportImportProject` and reference the source workspace connection configuration file, the project location, the target workspace connection configuration file, and the directory location in the target workspace where you want to import the files. If the source workspace uses a versioned repository connection type you do not have to check out the project files before you use the `exportImportProject` command to move your project.

You can also use the Blaze Advisor IDE to export and import a project, see:

- “Importing a Blaze Advisor Project from a Blaze Advisor Workspace or repository” in *DevelopingRuleProjects.pdf*
- “Exporting a Blaze Advisor Project to a Blaze Advisor Workspace” in *DevelopingRuleProjects.pdf*



Important You can also use the `exportImportProject` command to convert a project you have created in a workspace connected to one repository type to use another repository type. For example, you can export project contents from a workspace connected to a File repository and import them into a workspace connect to a Database repository. Likewise, assuming that the workspace types are the same, you can export project contents from a private workspace connected to a File BVS repository and import them into a local workspace connected to a CVS repository and check in the imported files.

Command, Argument, or Option	Description
exportImportProject	Command for exporting and importing a project. When you use this command, two actions occur. You export the contents of a project from a source directory and import the contents into a project in a target directory.
-sourceConnection	A pathname to the location of the source workspace connection file (.cfg). This file includes information such as the name of the repository connection, the location of the repository, and the name of the repository. See "Writing Connection Configuration Files" on page 169 .
-projectLocation	Argument takes the project directory location relative to the root directory in the source repository. You need to specify the Project folder name and the project name in the path. Example: "/ProjectA Folder/Project_A"
-targetConnection	A pathname to the target workspace location of the file (.cfg). This file includes information such as the name of the repository connection, the location of the repository, and the name of the repository. See "Writing Connection Configuration Files" on page 169 .
-directoryLocation	Argument takes a directory location relative to the root directory in the workspace where you want to import your project files. Example: "/ProjectB Folder"
-flat	Flatten subprojects while importing and exporting a project. See "Flattening Subprojects While Importing or Exporting Projects" in <i>DevelopingRuleProjects.pdf</i> .
-verbose	Option prints out all detailed messages into the command console.



Note Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, use double quotes.

Example:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil
exportImportProject
-sourceconnection "C:/Repo_Connections/FileWksp_A.cfg" -projectLocation "/Project_A Folder/Project_A" targetconnection "C:/Repo_Connections/
FileWksp_B.cfg"
-directoryLocation "/ProjectB Folder"-verbose
```



Importing a Workspace from a Zip File Using the Utility

You can import a workspace from a .zip file using the `NdRomAdminUtil` utility. To import a workspace from a .zip file, you use the command `importWorkspace` and reference the pathname to the .zip file whose contents you want to import and the pathname for connection configuration file for the target workspace or repository. You can also use the Blaze Advisor IDE, see “Importing a Blaze Advisor Contents from a .zip File” in *DevelopingRuleProjects.pdf*.

When the contents of the .zip file are imported to your workspace, the folders and files that are extracted from the .zip file are integrated with any existing folders and files in the workspace. If the workspace you are using is in a versioned repository, you need to check in the imported files.



Important If there are two entities with the same name, the entities in the .zip file will overwrite the contents of the entities in the current workspace.

Command, Argument, or Option	Description
<code>importWorkspace</code>	Command for importing a workspace from a .zip file.
<code>-workspaceConnection</code>	A pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See “ Writing Connection Configuration Files ” on page 169.
<code>-zipFilePath</code>	Argument takes a pathname to the .zip file whose contents you want to import into the workspace.
<code>-verbose</code>	Option prints out all detailed messages into the command console.



Note Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, use double quotes.

Example:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil importWorkspace  
-workspaceConnection "C:/Repo_Connections/FileWksp.cfg" -zipFilePath  
"C:/Blaze/Advisor/ZipFiles/WkspContents.zip" -verbose
```

Exporting a Workspace to a Zip File Using the Utility

You can export workspace contents to a .zip file using the `NdRomAdminUtil` utility. To export workspace contents to a .zip file, you use the command `exportWorkspace` and you reference the connection configuration file for the source workspace and the pathname to the .zip file where you want to export the contents. You can also use the

Blaze Advisor IDE, see “Exporting a Blaze Advisor Workspace to a .zip File” in *DevelopingRuleProjects.pdf*.

You can export the workspace contents as a .zip file so that other users can import these files into their workspace. When the folders and files of the .zip file are imported, they will be integrated with existing folders and files in the current workspace.

Command, Argument, or Option	Description
exportWorkspace	Command for exporting a workspace to a .zip file.
-workspaceConnection	A pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See “ Writing Connection Configuration Files ” on page 169.
-zipFilePath	Argument takes a pathname to the .zip file where you want to export the workspace contents.
-verbose	Option prints out all detailed messages into the command console.



Note Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, use double quotes.

Example:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil exportWorkspace
    -workspaceConnection "C:/Repo_Connections/FileWksp.cfg" -zipFilePath "C:/
    Blaze/Advisor/ZipFiles/WkspContents.zip" -verbose
```

Exporting and Importing a Workspace Using the Utility

You can export and import workspace contents from a source workspace or repository to a target workspace or repository using the `NdRomAdminUtil` utility.

You can also use the Blaze Advisor IDE, see:

- “Importing a Blaze Advisor Contents from Another Blaze Advisor Workspace” in *DevelopingRuleProjects.pdf*
- “Exporting a Blaze Advisor Contents to Another Blaze Advisor Workspace” in *DevelopingRuleProjects.pdf*

To export and import a workspace, you use the command `exportImportWorkspace` and reference the source workspace connection configuration file where you are exporting the contents and the target workspace connection configuration file where you want to import the contents. If the source workspace uses a versioned repository connection type, you do not have to check out the files in your workspace before you use the `exportImportWorkspace` command to move your workspace.



Important You can also use the `exportImportWorkspace` command to convert a workspace you have created using a given repository type to another repository type. For

example, you can export contents from a workspace connected to a File repository and import them into a file workspace connected to a Database repository or a MongoDB repository. Likewise, assuming that the workspace types are the same, you can export contents from a private workspace connected to a BVS repository and import them into a local workspace using a CVS repository connection and check them in.

Command, Argument, or Option	Description
exportImportWorkspace	Command for exporting and importing a workspace. When you use this command, two actions occur. You export the contents of a workspace from a source directory and import the contents into a workspace in a target directory.
-sourceConnection	A pathname to the location of the workspace connection file (.cfg). This file includes information such as the name of the repository connection, the location of the repository, and the name of the repository. See "Writing Connection Configuration Files" on page 169 .
-targetConnection	A pathname to the location of the target workspace location (.cfg). This file includes information such as the name of the repository connection, the location of the repository, and the name of the repository. See "Writing Connection Configuration Files" on page 169 .
-verbose	Option prints out all detailed messages into the command console.



Note Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, use double quotes.

Example:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil  
exportImportWorkspace  
-sourceconnection "C:/Repo_Connections/FileWksp_A.cfg" targetconnection  
"C:/Repo_Connections/FileWksp_B.cfg"-verbose
```

Releasing a Project Using the Utility

You can release a project to a directory in the same workspace or repository using the `releaseProject` command of the `NdRomAdminUtil` utility. When you release a project, you create a copy in the directory location of the workspace or repository you are currently using. You specify the location of the workspace or repository connection configuration file when you run the utility. You can also release a project in the Blaze Advisor IDE, see "Releasing Blaze Advisor Projects" in *DevelopingRuleProjects.pdf*.

You can also use the `NdRomAdminUtil` utility for the following related tasks:

- ["Setting a Release Manager" on page 78](#)
- ["Showing a Release Manager" on page 78](#)

This table describes the available arguments and options.

Command, Argument, or Option	Description
releaseProject	Command for releasing a project to another directory location in the workspace.
-workspaceConnection	A pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository, and the name of the repository. See " Writing Connection Configuration Files on page 169.
-projectLocation	Argument takes the project location relative to the root directory in the source repository. You need to specify the project folder name and the project name in the path. Example: "/ProjectA Folder/Project_A"
-directoryLocation	Argument takes a directory location relative to the root directory in the workspace where you want to release your project files. Example: "/ProjectB Folder"
-displayName	Argument takes the name for the released project. This is the name that is displayed in the Blaze Advisor IDE and in the Windows Explorer.
-m	Option uses a string value for a comment. Note Use double quotes around your comment text.
-flat	Flattens unreleased subprojects. See "Flattening Unreleased Subprojects While Releasing Projects" in <i>DevelopingRuleProjects.pdf</i> .
-verbose	Option prints out all detailed messages into the command console.



Note Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, use double quotes.



Note If you are using a versioned repository, note that the released project is not checked into the versioning system by default.

This is an example of a batch file used to release a project to a non-versioned repository:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil
exportImportProject
-sourceconnection "C:/Repo_Connections/FileWksp_A.cfg" -projectLocation "/Project_A Folder/Project_A" targetconnection "C:/Repo_Connections/FileWksp_B.cfg"
-directoryLocation "/ProjectB Folder"-verbose
```

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil releaseProject
-workspaceConnection "C:/Repo_Connections/FileWksp.cfg" -projectLocation "/
```

```
Project_A Folder/Project_A" -directoryLocation "/Rule Services/Project_A
Release 1"
-m "New released project" -verbose
```

Setting a Release Manager

You can set a custom release manager class for a repository using the `setReleaseManager` command of the `NdRomAdminUtil` utility. If you are using a versioned repository with private or local workspaces, you need to check in this change to the repository configuration.

In the batch file or command prompt, you reference the path to these files as argument values. See "["Running the NdRomAdminUtil Utility" on page 166](#)".

Command, Argument, or Option	Description
<code>setReleaseManager</code>	Command for adding a custom release manager class to a workspace.
<code>-workspaceConnection</code>	Argument takes a pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See " "Writing Connection Configuration Files" on page 169 ". Alternatively, you can use a symbolic name for the workspace connection. See " "Entering a Symbolic Name or a Pathname as an Argument Value" on page 166 ".
<code>-releaseFactory</code>	Argument takes a fully qualified name of a release manager class.
<code>-verbose</code>	Option prints out all detailed messages into the command console.

Example of a batch file used to add a custom query class to a workspace:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil addQuery
-workspaceConnection "c:/Repo_Connections/ConnectionFiles/
wkspConnection.cfg" -queryFactory com.mypackage.myReleaseMgr -verbose
```

Showing a Release Manager

You can display the name of the release manager class for a repository using the `showReleaseManager` command of the `NdRomAdminUtil` utility.

In the batch file or command prompt, you reference the path to these files as argument values. See "["Running the NdRomAdminUtil Utility" on page 166](#)".

Command, Argument, or Option	Description
showReleaseManager	Command for displaying a Release Manager class name.
-workspaceConnection	<p>Argument takes a pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See "Writing Connection Configuration Files" on page 169.</p> <p>Alternatively, you can use a symbolic name for the workspace connection. See "Entering a Symbolic Name or a Pathname as an Argument Value" on page 166.</p>
-verbose	Option prints out all detailed messages into the command console.

Example of a batch file used to show a Release Manager class in a repository configuration:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil
showReleaseManager
-workspaceConnection "c:/Repo_Connections/ConnectionFiles/
wkspConnection.cfg" -verbose
```

Publishing a Project Using the Utility

You can publish a project to a directory in a different workspace or repository using the `publishProject` command of the `NdRomAdminUtil` utility. When you publish a project you create a copy in the directory location of a different workspace or repository. You specify the location of the connection configuration file when you run the utility. You can also use the Blaze Advisor IDE, see "Publishing Blaze Advisor Projects" in *DevelopingRuleProjects.pdf*.

You can also use the `NdRomAdminUtil` utility for the following related tasks:

- ["Updating a Published Project Using the Utility" on page 81](#)
- ["Replacing a Published Project Using the Utility" on page 82](#)

This table describes the available arguments and options.

Command, Argument, or Option	Description
publishProject	Command for publishing a project to another directory location in the workspace.
-sourceConnection	<p>A pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository, and the name of the repository. See "Writing Connection Configuration Files" on page 169.</p>

Command, Argument, or Option	Description
-projectLocation	Argument takes the project location relative to the root directory in the source repository. You need to specify the project folder name and the project name in the path. Example: "/ProjectA Folder/Project_A"
-targetConnection	A pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository, and the name of the repository. See " Writing Connection Configuration Files " on page 169.
-directoryLocation	Argument takes a directory location relative to the root directory in the workspace where you want to publish your project. Example: "/ProjectB Folder"
-displayName	Argument takes the name for the published project. This is the name that is displayed in the Blaze Advisor IDE and in the Windows Explorer.
-m	Takes a string value to input a comment. Note Use double quotes around your comment text.
-flat	Flattens target subprojects. See "Flattened Target Option" in <i>DevelopingRuleProjects.pdf</i> .
-subProjectProcess	Option to replace or update a subproject that has been previously published.
verbose	Prints out detailed messages in the command console

 **Note** Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, use double quotes.

This is an example of a batch file used to publish a project to a non-versioned repository:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil
exportImportProject
-sourceConnection "C:/Repo_Connections/FileWksp_A.cfg" -projectLocation "/Project_A Folder/Project_A" targetconnection "C:/Repo_Connections/FileWksp_B.cfg"
-directoryLocation "/ProjectB Folder"-verbose

java com.blazesoft.template.repository.admin.NdRomAdminUtil publishProject
-sourceConnection "C:/Repo_Connections/FileWksp_A.cfg" -projectLocation "/Project_A Folder/Project_A" -targetConnection "C:/Repo_Connections/FileWksp_B.cfg"
-directoryLocation "/Rule Services/Project_A"-displayName "Publish 1"
-m "New published project" -verbose
```

Updating a Published Project Using the Utility

You can update a project that you previously published to a directory in a different repository using the `updatePublishedProject` command of the `NdRomAdminUtil` utility. You update a published project when you want changes you made to the source project to be reflected in the published version.

This table describes the available arguments and options.

Command, Argument, or Option	Description
<code>updatePublishedProject</code>	Command for updating a published project
<code>-sourceConnection</code>	A pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository, and the name of the repository. See " Writing Connection Configuration Files " on page 169.
<code>-projectLocation</code>	Argument takes the project location relative to the root directory in the source repository. You need to specify the project folder name and the project name in the path. Example: "/ProjectA Folder/Project_A"
<code>-targetConnection</code>	A pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository, and the name of the repository. See " Writing Connection Configuration Files " on page 169.
<code>-targetProjectLocation</code>	Argument takes a directory location relative to the root directory in the workspace where you want to update the published project. Example: "/ProjectB Folder"
<code>-m</code>	Takes a string value to input a comment. Note Use double quotes around your comment text.
<code>-verbose</code>	Prints out detailed messages in the command console
<code>-revertOnError</code>	If an error occurs during the update operation, it reverts the target published project to its original state.
<code>-flat</code>	Flattens target subprojects. See " Flattened Target Option " in <i>DevelopingRuleProjects.pdf</i> .



Note Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, use double quotes.

This is an example of a batch file used to update a published project in a non-versioned repository:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil  
exportImportProject  
-sourceConnection "C:/Repo_Connections/FileWksp_A.cfg" -projectLocation "/  
Project_A Folder/Project_A" targetconnection "C:/Repo_Connections/  
FileWksp_B.cfg"  
-directoryLocation "/ProjectB Folder"-verbose  
java com.blazesoft.template.repository.admin.NdRomAdminUtil  
updatePublishedProject  
-sourceConnection "C:/Repo_Connections/FileWksp_A.cfg" -projectLocation  
"/Project_A Folder/Project_A" -targetConnection "C:/Repo_Connections/  
FileWksp_B.cfg"  
-targetProjectLocation "/Rule Services/Project_A" -m "updated published  
project"  
-verbose -revertOnError
```

Replacing a Published Project Using the Utility

You can replace project you previously published with a new version by using the `replacePublishedProject` command of the `NdRomAdminUtil` utility.

This table describes the available arguments and options.

Command, Argument, or Option	Description
<code>replacePublishedProject</code>	Command for replacing a published project
<code>-sourceConnection</code>	A pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository, and the name of the repository. See " Writing Connection Configuration Files " on page 169.
<code>-projectLocation</code>	Argument takes the project location relative to the root directory in the source repository. You need to specify the project folder name and the project name in the path. Example: "/ProjectA Folder/Project_A"
<code>-targetConnection</code>	A pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository, and the name of the repository. See " Writing Connection Configuration Files " on page 169.
<code>-targetProjectLocation</code>	Argument takes a location relative to the root directory in the workspace where the published project you want to replace exists. Example: "/ProjectB Folder"
<code>-m</code>	Takes a string value to input a comment. Note Use double quotes around your comment text.

Command, Argument, or Option	Description
-verbose	Prints out detailed messages in the command console
-revertOnError	If an error occurs during the replacement operation, it reverts the target project to its original state.
-flat	Flattens target subprojects. See “Flattened Target Option” in <i>DevelopingRuleProjects.pdf</i> .



Note Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, use double quotes.

This is an example of a batch file used to replace a published project to a non-versioned repository:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil
exportImportProject
-sourceconnection "C:/Repo_Connections/FileWksp_A.cfg" -projectLocation "/Project_A Folder/Project_A" targetconnection "C:/Repo_Connections/FileWksp_B.cfg"
-directoryLocation "/ProjectB Folder"-verbose
java com.blazesoft.template.repository.admin.NdRomAdminUtil
replacePublishedProject
-sourceConnection "C:/Repo_Connections/FileWksp_A.cfg" -projectLocation
"/Project_A Folder/Project_A" -targetConnection "C:/Repo_Connections/
FileWksp_B.cfg"
-targetProjectLocation "/Rule Services/Project_A" -m "replaced published
project"
-verbose -revertOnError
```

Managing Versioned Project Content Using the Utility

This section describes how to manage the version history of project files and project items using a repository with Blaze versioning BVS, or local CVS or local Subversion or ClearCase versioning services. You can manage the version history using the Blaze Advisor IDE or the `NdRomAdminUtil` utility. Where applicable, links to similar topics using the Blaze Advisor IDE are included in the following topics:

- “About Managing Projects, Project Items and Folders Using a Versioning Service” in *DevelopingRuleProjects.pdf*
- “Checking In New or Existing Items Using the Utility” on page 84
- “Checking Out One or More Items Using the Utility” on page 85
- “Canceling the Check Out for One or More Items Using the Utility” on page 86
- “Deleting a Versioned Item Using the Utility” on page 87
- “Restoring Logically Deleted Items” on page 88
- “Updating a Private or Local Workspace Using the Utility” on page 91

- “Viewing Item History Using the Utility” on page 89
- “Checking the Status of Workspace Files Using the Utility” on page 90
- “Promoting an Item Using the Utility” on page 91
- “Adding Labels to a Workspace Entry” on page 92
- “Retrieving Labels for a Workspace Entry” on page 93
- “Deleting a Label from a Workspace Entry” on page 94

For information about versioning operations for folders, see “Working with Folders in a Versioned Repository” in *DevelopingRuleProjects.pdf*.

Checking In New or Existing Items Using the Utility

You can check in new or modified existing item individually or check in the contents of a folder using the `NdRomAdminUtil` utility. To check in new or existing items from your workspace into the repository, use the `checkIn` command, then reference the connection configuration file for the workspace or repository, and specify the directory location and name of the file, if applicable, to check in.

You can also use the equivalent command in the Blaze Advisor IDE, see “Checking In New or Existing Items in the Blaze Advisor IDE” in *DevelopingRuleProjects.pdf*.

Command, Argument, or Option	Description
<code>checkIn</code>	Command for checking in one or more items in a directory.
<code>-workspaceConnection</code>	Argument takes a path name to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See “ Writing Connection Configuration Files ” on page 169. Alternatively, you can use a symbolic name for the workspace connection. See “ Entering a Symbolic Name or a Pathname as an Argument Value ” on page 166.
<code>-entryLocation</code>	Argument takes a directory or item location relative to the root directory. If you were checking in an item in a folder: “/Technical Library Folder/enumeration1” If you were checking in a project file: “/Project Name” If you were checking in the contents of a folder: “/Technical Library Folder”
<code>-l</code>	Option executes only in the local directory. If you do not use this option, the <code>checkIn</code> command executes recursively through the whole workspace.

Command, Argument, or Option	Description
-verbose	Option prints out all detailed messages into the command console.
-m	Option takes a string value for a comment when you check in an item. Note Use double quotes around comment text.



Note Whenever you enter path names or the names of files with spaces in a command prompt or in a batch file, we recommend that you use double quotes.

Example of a batch file to check in an individual item in a folder:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil checkIn
    -workspaceConnection "C:/Repo_Connections/FileWksp.cfg" -entryLocation
        "/Data Validation Ruleflow Testing Folder/class1" -m "New class for testing
        purposes." -verbose
```

Checking Out One or More Items Using the Utility

You can check out one or more items using the `NdRomAdminUtil` utility. To check out one or more items from your repository, use the `checkOut` command, then reference the connection configuration file for the workspace or repository, and specify the directory location and name of the file, if applicable, to check out.

You can also use the equivalent command in the Blaze Advisor IDE, see “Checking Out One or More Items in the Blaze Advisor IDE” in *DevelopingRuleProjects.pdf*.

Command, Argument, or Option	Description
<code>checkOut</code>	Command for checking out one or more items from a directory.
<code>-workspaceConnection</code>	Argument takes a pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See “ Writing Connection Configuration Files ” on page 169. Alternatively, you can use a symbolic name for the workspace connection. See “ Entering a Symbolic Name or a Pathname as an Argument Value ” on page 166.
<code>-entryLocation</code>	Argument takes a directory or item location relative to the root directory. If you were checking in an item in a folder: “/Technical Library Folder/enumeration1” If you were checking in a project file: “/Project Name” If you were checking in the contents of a folder: “/Technical Library Folder”

Command, Argument, or Option	Description
-l	Option executes only in the local directory. If you do not use this option, the checkIn command executes recursively through the whole workspace.
-verbose	Option prints out all detailed messages into the command console.

 **Note** Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, we recommend that you use double quotes.

Example of a batch file to check out an individual item in a folder:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil checkOut  
-workspaceConnection "C:/Repo_Connections/FileWksp.cfg" -entryLocation  
"/Data Validation Ruleflow Testing Folder/class1" -verbose
```

Cancelling the Check Out for One or More Items Using the Utility

You can cancel the check out of one or more items using the `NdRomAdminUtil` utility. To cancel the check out of one or more items from your repository, use the `cancelCheckOut` command, then reference the connection configuration file for the workspace or repository, and specify the directory location and name of the file, if applicable, to cancel the check out.

You can also use the equivalent command in the Blaze Advisor IDE, see “Canceling a Check Out for One or More Items in the Blaze Advisor IDE” in *DevelopingRuleProjects.pdf*

Command, Argument, or Option	Description
<code>cancelCheckOut</code>	Command for canceling the check out for one or more items from a directory.
<code>-workspaceConnection</code>	Argument takes a pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See “ Writing Connection Configuration Files ” on page 169. Alternatively, you can use a symbolic name for the workspace connection. See “ Entering a Symbolic Name or a Pathname as an Argument Value ” on page 166.
<code>-entryLocation</code>	Argument takes a directory or item location relative to the root directory. If you were checking in an item in a folder: “/Technical Library Folder/enumeration1” If you were checking in a project file: “/Project Name” If you were checking in the contents of a folder: “/Technical Library Folder”

Command, Argument, or Option	Description
-l	Optional executes only in the local directory. If you do not use this option, the checkIn command executes recursively through the whole workspace.
-verbose	Option prints out all detailed messages into the command console.



Note Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, we recommend that you use double quotes.

Example of a batch file to cancel check out an individual item in a folder:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil cancelCheckOut
-workspaceConnection "C:/Repo_Connections/FileWksp.cfg" -entryLocation
"/Data Validation Ruleflow Testing Folder/class1" -verbose
```

Deleting a Versioned Item Using the Utility

You can delete a versioned project or project item using the `NdRomAdminUtil` utility. However, you cannot delete a folder from a versioned repository. See "Working with Folders in a Versioned Repository" in *DevelopingRuleProjects.pdf*.

To delete a versioned item from your repository, use the `delete` command, then reference the connection configuration file for the workspace or repository, and specify the directory location and name of the file to delete. Prior to using the `delete` command, you need to check out the item using the utility or Blaze Advisor IDE. After using the `delete` command you need to check in the item using the utility or the Blaze Advisor IDE.

You can also use the equivalent command in the Blaze Advisor IDE, see "Deleting Versioned Items in the Blaze Advisor IDE" in *DevelopingRuleProjects.pdf*.



Note You cannot restore a versioned item using the utility. Use the Restore command in the IDE. See "Restoring Deleted Items in the Blaze Advisor IDE" in *DevelopingRuleProjects.pdf*.



Caution If you are using an Clearcase repository connected to an IBM Rational ClearCase server, you cannot restore a deleted item. If you check out an item and delete it, you can Cancel Check out but you cannot restore it after you check in your changes.

Command, Argument, or Option	Description
delete	Command for deleting a versioned item from a directory.
-workspaceConnection	Argument takes a pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See “ Writing Connection Configuration Files ” on page 169. Alternatively, you can use a symbolic name for the workspace connection. See “ Entering a Symbolic Name or a Pathname as an Argument Value ” on page 166.
-entryLocation	Argument takes a directory or item location relative to the root directory. If you were checking in an item in a folder: “/Technical Library Folder/enumeration1” If you were checking in a project file: “/Project Name” If you were checking in the contents of a folder: “/Technical Library Folder”
-l	Option executes only in the local directory. If you do not use this option, the checkIn command executes recursively through the whole workspace.
-verbose	Option prints out all detailed messages into the command console.



Note Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, we recommend that you use double quotes.

Example of a batch file to delete a versioned item from a folder:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil delete  
-workspaceConnection "C:/Repo_Connections/FileWksp.cfg" -entryLocation  
"/Data Validation Ruleflow Testing Folder/class1" -verbose
```

Restoring Logically Deleted Items

You can restore a logically deleted item or items in a directory using the `NdRomAdminUtil` utility. To restore an item, use the `restore` command, then reference the connection configuration file for the workspace or repository, and specify the directory location and name of the file, if applicable, to restore.

You can also use the equivalent command in the Blaze Advisor IDE, see “Restoring Deleted Items in the Blaze Advisor IDE” in *DevelopingRuleProjects.pdf*.

Command, Argument, or Option	Description
delete	Command for deleting a versioned item from a directory.
-workspaceConnection	<p>Argument takes a pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See “Writing Connection Configuration Files” on page 169.</p> <p>Alternatively, you can use a symbolic name for the workspace connection. See “Entering a Symbolic Name or a Pathname as an Argument Value” on page 166.</p>
-entryLocation	<p>Argument takes a directory or item location relative to the root directory.</p> <p>If you were checking in an item in a folder: “/Technical Library Folder/enumeration1”</p> <p>If you were checking in a project file: “/Project Name”</p> <p>If you were checking in the contents of a folder: “/Technical Library Folder”</p>
-l	Option executes only in the local directory. If you do not use this option, the checkIn command executes recursively through the whole workspace.
-verbose	Option prints out all detailed messages into the command console.



Note Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, we recommend that you use double quotes.

Example of a batch file to restore a logically deleted item:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil restore
-workspaceConnection "C:/Repo_Connections/FileWksp.cfg" -entryLocation
"/Data Validation Ruleflow Testing Folder/class1" -verbose
```

Viewing Item History Using the Utility

You can view the version history of an item or items in a directory using the NdRomAdminUtil utility. To view the item history, use the `history` command, then reference an XML file for the workspace connection configuration, and specify the directory location and name of the file, if applicable, to view.

You can also use the equivalent command in the Blaze Advisor IDE, see “Viewing Item History in the Blaze Advisor IDE” in *DevelopingRuleProjects.pdf*.

Command, Argument, or Option	Description
history	Command for viewing the version history of an item or items in a directory.
-workspaceConnection	Argument takes a pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See “ Writing Connection Configuration Files ” on page 169. Alternatively, you can use a symbolic name for the workspace connection. See “ Entering a Symbolic Name or a Pathname as an Argument Value ” on page 166.
-entryLocation	Argument takes a directory or item location relative to the root directory. If you were checking in an item in a folder: “/Technical Library Folder/enumeration1” If you were checking in a project file: “/Project Name” If you were checking in the contents of a folder: “/Technical Library Folder”
-l	Option executes only in the local directory. If you do not use this option, the checkIn command executes recursively through the whole workspace.
-verbose	Option prints out all detailed messages into the command console.



Note Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, we recommend that you use double quotes.

Example of a batch file to view the history of an item in a folder:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil history  
-workspaceConnection "C:/Repo_Connections/FileWksp.cfg" -entryLocation  
"/Data Validation Ruleflow Testing/class1" -verbose
```

Checking the Status of Workspace Files Using the Utility

You can check the status of files in a workspace using the NdRomAdminUtil utility. To check the status of workspace files, use the `status` command, then reference an XML file for the connection to the workspace, and specify the directory location and name of the file, if applicable, to update.

You can also use the equivalent command in the Blaze Advisor IDE, see “[Checking the Status of Workspace Files in the Blaze Advisor IDE](#)” in *DevelopingRuleProjects.pdf*.

Command, Argument, or Option	Description
status	Command for checking the status of workspace files.
-workspaceConnection	<p>Argument takes a pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See “Writing Connection Configuration Files” on page 169.</p> <p>Alternatively, you can use a symbolic name for the workspace connection. See “Entering a Symbolic Name or a Pathname as an Argument Value” on page 166.</p>
-entryLocation	<p>Argument takes a directory or item location relative to the root directory.</p> <p>If you were checking in an item in a folder: “/Technical Library Folder/enumeration1”</p> <p>If you were checking in a project file: “/Project Name”</p> <p>If you were checking in the contents of a folder: “/Technical Library Folder”</p>
-l	Option executes only in the local directory. If you do not use this option, the checkIn command executes recursively through the whole workspace.
-verbose	Option prints out all detailed messages into the command console.



Note Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, we recommend that you use double quotes.

Example of a batch file to check the status of workspace files:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil status
(workspaceConnection "C:/Repo_Connections/FileWksp.cfg" -entryLocation
"/Data Validation Ruleflow Testing/class1" -verbose
```

Promoting an Item Using the Utility

Currently, there is no NdRomAdminUtil utility command to promote an item. If you want to promote a revision, you need to use the IDE. See “Promoting a Previous Version of an Item in the Blaze Advisor IDE” in *DevelopingRuleProjects.pdf*.

Updating a Private or Local Workspace Using the Utility

The Update command overwrites the files in your private BVS, or local CVS, or local Subversion or ClearCase workspace with the latest version of files from the versioning service. You need to periodically update the files in your private workspace to get the latest versions of files checked in by other users. You can update the repository, a folder,

or the individual items in a project folder. If you choose to update the repository, all of the items are updated recursively. You can use one of the following methods:

You can update an individual item or items in a directory of a private or local workspace using the `NdRomAdminUtil` utility. To update items, use the `update` command, then reference an XML file for the workspace connection configuration, and specify the directory location and name of the file, if applicable, to update.

Command, Argument, or Option	Description
<code>update</code>	Command for updating an item or items in a directory.
<code>-workspaceConnection</code>	Argument takes a pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See “ Writing Connection Configuration Files ” on page 169. Alternatively, you can use a symbolic name for the workspace connection. See “ Entering a Symbolic Name or a Pathname as an Argument Value ” on page 166.
<code>-entryLocation</code>	Argument takes a directory or item location relative to the root directory. If you were checking in an item in a folder: “/Technical Library Folder/enumeration1” If you were checking in a project file: “/Project Name” If you were checking in the contents of a folder: “/Technical Library Folder”
<code>-l</code>	Option executes only in the local directory. If you do not use this option, the <code>checkIn</code> command executes recursively through the whole workspace.
<code>-verbose</code>	Option prints out all detailed messages into the command console.



Note Whenever you enter pathnames or the names of files with spaces in a command prompt or in a batch file, we recommend that you use double quotes.

Example of a batch file to update an individual item in a folder:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil update  
-workspaceConnection "C:/Repo_Connections/FileWksp.cfg" -entryLocation  
"/Data Validation Ruleflow Testing/class1" -verbose
```

Adding Labels to a Workspace Entry

If you are using a BVS or CVS repository and want to add a label to categorize your workspace entries, you can run the `NdRomAdminUtil` utility `label` command.

In the batch file or command prompt, you reference the path to these files as argument values. See ["Running the NdRomAdminUtil Utility" on page 166](#).

Command, Argument, or Option	Description
label	Command for adding a label to a workspace entry.
-workspaceConnection	Argument takes a pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See "Writing Connection Configuration Files" on page 169 . Alternatively, you can use a symbolic name for the workspace connection. See "Entering a Symbolic Name or a Pathname as an Argument Value" on page 166 .
-entryLocation	Argument takes the relative location of the folder within the workspace where you want to add the label.
-label	Argument takes the label name you want to add to the folder.
-l	Option to limit the labelling to the local folder with no recursion.
-verbose	Option prints out all detailed messages into the command console.

Example of a batch file used to add a label to a workspace entry:

```
<ADVISOR_HOME>/setenv
java com.blazesoft.template.repository.admin.NdRomAdminUtil label
    -workspaceConnection "c:/Repo_Connections/ConnectionFiles/
        wkspConnection.cfg" -entryLocation "/MyProject1 Folder" -label DoNextRelease
    -verbose
```

Retrieving Labels for a Workspace Entry

If you are using a BVS or CVS repository and want to see a list of all labels for a workspace entry, you can run the NdRomAdminUtil utility `getLabels` command.

In the batch file or command prompt, you reference the path to these files as argument values. See ["Running the NdRomAdminUtil Utility" on page 166](#).

Command, Argument, or Option	Description
getLabel	Command for obtaining a list of labels for a workspace entry.
-workspaceConnection	Argument takes a pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See " Writing Connection Configuration Files " on page 169. Alternatively, you can use a symbolic name for the workspace connection. See " Entering a Symbolic Name or a Pathname as an Argument Value " on page 166.
-entryLocation	Argument takes the relative location of the folder within the workspace where you want to retrieve the list of labels. The getLabels command is limited to the target workspace entry you specify with this argument.
-verbose	Option prints out all detailed messages into the command console.

Example of a batch file used to get all labels for a workspace entry:

```
<ADVISOR_HOME>/setenv  
java com.blazesoft.template.repository.admin.NdRomAdminUtil getLabels  
-workspaceConnection "c:/Repo_Connections/ConnectionFiles/  
wkspConnection.cfg" -entryLocation "/MyProject1 Folder" -verbose
```

Deleting a Label from a Workspace Entry

If you are using a BVS or CVS repository and want to delete a label from your workspace entries, you can use the you can run the NdRomAdminUtil utility `deleteLabel` command.

In the batch file or command prompt, you reference the path to these files as argument values. See "[Running the NdRomAdminUtil Utility](#)" on page 166.

Command, Argument, or Option	Description
deleteLabel	Command for deleting a label from a workspace entry.
-workspaceConnection	Argument takes a pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See " Writing Connection Configuration Files " on page 169. Alternatively, you can use a symbolic name for the workspace connection. See " Entering a Symbolic Name or a Pathname as an Argument Value " on page 166.

Command, Argument, or Option	Description
-entryLocation	Argument takes the relative location of the folder within the workspace where you want to delete the label.
-label	Argument takes the label name you want to delete from the folder.
-l	Option to limit the deletion to the local folder with no recursion.
-verbose	Option prints out all detailed messages into the command console.

Example of a batch file used to delete a label from a workspace entry:

```
<ADVISOR_HOME>/setenv
java com.blazesoft.template.repository.admin.NdRomAdminUtil deleteLabel
    -workspaceConnection "c:/Repo_Connections/ConnectionFiles/
    wkspConnection.cfg" -entryLocation "/MyProject1 Folder" -label DoNextRelease
    -verbose
```

Managing Files Using the Utility

You can perform a number of administrative tasks that allow you obtain information from a workspace, unlock files, and can the repository configuration.

If you are using a BVS repository, you can obtain a list of locked files and unlock files them using the **NdRomAdminUtil** utility.

- “[Obtaining a List of Locked Files](#)” on page 95
- “[Unlocking Files by User](#)” on page 96
- You can change your repository configuration by adding or remove a custom query class and you can obtain a list of custom query classes in your repository. “[Adding a Query](#)” on page 97
- “[Removing a Query](#)” on page 98
- “[Listing Queries](#)” on page 99
- “[Applying Fixes](#)” on page 99

Obtaining a List of Locked Files

If you are using a BVS repository, you can run the **NdRomAdminUtil** utility to obtain a list of locked file. To run the utility, you need to reference the connection configuration file you wrote to create the workspace or repository. If you did not create the workspace or repository using the utility, you need to write a connection configuration file that specifies the workspace or repository connection parameters and the connection name. For information what needs to be in a connection file, see “[Repository Connection Configuration File Examples](#)” on page 171.

-  **Note** If you are using a CVS, SVN, or Clearcase repository, you cannot obtain a list of locked files using the `NdRomAdminUtil` utility. The ability to list locks or release checked out entries in external SCMs is not supported in Blaze Advisor. To obtain a list of locked entries CVS, SVN, or Clearcase consult the documentation for the external SCM.

In the batch file or command prompt, you reference the path to these files as argument values. See "["Running the NdRomAdminUtil Utility" on page 166](#)".

Command, Argument, or Option	Description
<code>listLocks</code>	Command to obtain a list of locked files
<code>-repositoryConnection</code>	Argument takes the path to the location of the repository connection information(.cfg). See " "Repository Connection Configuration File Examples" on page 171 ". This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
<code>-verbose</code>	Option prints out all details messages into the command console.

Example of a batch file used to obtain a list of locks from a File BVS repository:
`java com.blazesoftware.template.repository.admin.NdRomAdminUtil listLocks
-repositoryConnection "c:/Repo_Connections/ConnectionFiles/
BVSSharedRepo.cfg"
-verbose`

Unlocking Files by User

If you are using a BVS repository, you can run the `NdRomAdminUtil` utility to release a lock for a file or files locked by a specific user. To run the utility, you need to reference the connection file you wrote to create a repository. If you did not create the repository using the utility, you need to write a connection file that specifies the repository connection parameters and the connection name. See "["Repository Connection Configuration File Examples" on page 171](#)".

-  **Note** If you are using a CVS, SVN, or Clearcase repository, you cannot obtain a list of locked files using the `NdRomAdminUtil` utility. The ability to list locks or release checked out entries in external SCMs is not supported in Blaze Advisor. To obtain a list of locked entries CVS, SVN, or Clearcase consult the documentation for the external SCM.

You can release locks in the RMA with LCM Access Control if you have the RMA Administrator role and the project used to generate the RMA is stored in a BVS repository. See "["Releasing Locks" in *DevelopingRuleMaintenanceApplications.pdf*](#)".

In the batch file or command prompt, you reference the path to these files as argument values. See "["Running the NdRomAdminUtil Utility" on page 166](#)".

Command, Argument, or Option	Description
releaseLocksByUser	Command release locks on files checked out by a specific user.
-repositoryConnection	Argument takes the path to the location of the repository connection information(.cfg). See "Repository Connection Configuration File Examples" on page 171 . This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
-directoryLocation	Argument takes a directory and or project location relative to the root directory where you want to release the locks.
-user	Argument takes a case-sensitive user name for the value. The user who has locked the files you want to release.
-verbose	Option prints out all details messages into the command console.

Example of a batch file used to obtain release files checked out by a specific user from a File BVS repository:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil
releaseLocksByUser
-repositoryConnection "c:/Repo_Connections/ConnectionFiles/
BVSSharedRepo.cfg"
-directoryLocation "/Project_A" -user "Template Developer"-verbose
```

Adding a Query

If you want to add a custom ROM query class to your workspace after you have created your repository, you can run the `NdRomAdminUtil` utility `addQuery` command. Run the utility for each custom query you want to add to your repository. The custom query classes are added to the repository configuration. If you are using a versioned repository with private or local workspaces, you need to check in this change to the repository configuration.

In the batch file or command prompt, you reference the path to these files as argument values. See ["Running the NdRomAdminUtil Utility" on page 166](#).

Command, Argument, or Option	Description
addQuery	Command for adding a custom query class to a workspace.
-workspaceConnection	Argument takes a pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See “Writing Connection Configuration Files” on page 169 . Alternatively, you can use a symbolic name for the workspace connection. See “Entering a Symbolic Name or a Pathname as an Argument Value” on page 166 .
-queryFactory	Argument takes a fully qualified name of a custom query class.
-verbose	Option prints out all detailed messages into the command console.

Example of a batch file used to add a custom query class to a workspace:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil addQuery  
-workspaceConnection "c:/Repo_Connections/ConnectionFiles/  
wkspConnection.cfg" -queryFactory com.mypackage.mycustomQuery -verbose
```

Removing a Query

If you want to remove a custom ROM query class from your workspace after you have created your repository, you can run the `NdRomAdminUtil` utility `removeQuery` command. Run the utility for each custom query you want to remove from your repository. The custom query classes are removed from your repository configuration. If you are using a versioned repository with private or local workspaces, you need to check in this change to the repository configuration.

In the batch file or command prompt, you reference the path to these files as argument values. See [“Running the NdRomAdminUtil Utility” on page 166](#).

Command, Argument, or Option	Description
removeQuery	Command for removing a custom query class from a workspace.
-workspaceConnection	Argument takes a pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See “Writing Connection Configuration Files” on page 169 . Alternatively, you can use a symbolic name for the workspace connection. See “Entering a Symbolic Name or a Pathname as an Argument Value” on page 166 .

Command, Argument, or Option	Description
-queryFactory	Argument takes a fully qualified name of a custom query class.
-verbose	Option prints out all detailed messages into the command console.

Example of a batch file used to remove a custom query class from a workspace:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil removeQuery
    -workspaceConnection "c:/Repo_Connections/ConnectionFiles/
    wkspConnection.cfg" -queryFactory com.mypackage.mycustomQuery -verbose
```

Listing Queries

If you want to list the custom query classes in your repository configuration, you can run the `NdRomAdminUtil` utility `listQueries` command.

In the batch file or command prompt, you reference the path to these files as argument values. See ["Running the NdRomAdminUtil Utility" on page 166](#).

Command, Argument, or Option	Description
listQueries	Command for listing all custom query classes in a repository configuration
-workspaceConnection	Argument takes a pathname to the location of the workspace connection file (.cfg). The connection file includes information such as the name of the repository connection, the location of the repository and the name of the repository. See "Writing Connection Configuration Files" on page 169 . Alternatively, you can use a symbolic name for the workspace connection. See "Entering a Symbolic Name or a Pathname as an Argument Value" on page 166 .
-verbose	Option prints out all detailed messages into the command console.

Example of a batch file used to list all configured queries in a workspace:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil listQueries
    -workspaceConnection "c:/Repo_Connections/ConnectionFiles/
    wkspConnection.cfg" -verbose
```

Applying Fixes

If you want ensure that all instances in your workspace are up-to-date if the templates have changed, you can run the `NdRomAdminUtil` utility `applyFix` command to analyze your workspace contents, identify any out-of-date XML, and replace those files with the latest files using the configuration file

`com\blazesoft\template\repository\admin\default_rom_admin_config.cfg` supplied by

Blaze Advisor when you run the utility. Alternatively, you can fix the instance files by adding the last checked in attributes.

The `NdInstanceFixerExtension` in the `NdRomAdminUtil` utility `applyFix` command allows instances to be updated in cases where a value holder is added or removed in the referenced template. The `-projectLocation` optional argument and the command option `-autocheckin` are available for use with this extension. (Most fixes to metaphor instances, other than the addition or removal of value holders in the metaphor template are not supported by this extension.)

In a batch file or command prompt, you reference the path to these files as argument values. See ["Running the NdRomAdminUtil Utility" on page 166](#).

Command, Argument, or Option	Description
applyFix	Command for apply fixes to a workspace.
-workspaceConnection	Argument takes a pathname to the location of the workspace connection file (.cfg). You write the connection file and include information such as the name of the repository connection, the location of the repository and the name of the repository. The <code>NdRomAdminUtil</code> utility uses the <code>default_rom_admin_config.cfg</code> to update all instances located in the workspace you specify. See "Writing Connection Configuration Files" on page 169 . Alternatively, you can use a symbolic name for the workspace connection. See "Entering a Symbolic Name or a Pathname as an Argument Value" on page 166 .
-addLastCheckInAttributes	Argument updates the instances in your workspace based on the last check in attributes and ignores the configuration file <code>default_rom_admin_config.cfg</code> associated with the <code>NdRomAdminUtil</code> installed with Blaze Advisor.
-projectLocation	Allows you to specify the location of the project to be updated in the workspace connection file.
-autoCheckin	If used with projects in a versioned repository, checks out files that need to be updated, fixes them and checks in the changes.
-verbose	Option prints out all detailed messages into the command console.

Example of a batch file used to update the instance files in a given workspace based on the `default_rom_admin_config.cfg` configuration file supplied by Blaze Advisor to be used by the utility:

```
<ADVISOR_HOME>/setenv  
java com.blazesoft.template.repository.admin.NdRomAdminUtil applyFix  
-workspaceConnection "c:/Repo_Connections/ConnectionFiles/  
wkspConnection.cfg" -projectLocation "/Project1 Folder/Project1 Project"  
-autoCheckIn -verbose
```

Example of a batch file used to update the instance files in a given workspace using the last check in attributes:

```
<ADVISOR_HOME>/setenv  
java com.blazesoft.template.repository.admin.NdRomAdminUtil applyFix  
-workspaceConnection "c:/Repo_Connections/ConnectionFiles/  
wkspConnection.cfg" -addLastCheckinAttributes -verbose
```



Note If you use the `-addLastCheckinAttributes` option, the utility ignores the configuration file supplied by Blaze Advisor.

Adding Management Properties to a Repository

You can add management properties to a repository and use them to retrieve and display information about the items in your projects.



Note When you select a project to edit in the Blaze Advisor IDE, you can add management properties to the workspace or repository where the files for this project are stored.

This topic contains the following sections:

- “Management Properties” on page 101
- “Predefined Management Properties” on page 105
- “Adding a Management Property to an Entity Category” on page 102
- “Creating a Management Property” on page 110
- “Displaying Management Properties in a Folder or Project Editor” on page 117
- “Cutting, Copying, and Pasting Project Items with Management Properties” on page 118

In some cases, management properties are used to designate certain functions or rulesets as test cases for use with the brUnit module. See “Unit Testing Rule Projects” in *DevelopingRuleProjects.pdf*. For information about the Test Role and Test Case management properties, see “Entity Categories” on page 109.

Management Properties

Management properties provide a way to capture additional information about a project item, such as the date it was originally created or the date it was last modified. Each repository has access to a predefined set of management properties that are installed with Blaze Advisor. You can also create custom management properties to retrieve specific types of information.

Depending on the types of information you want to gather on your project items, you add the predefined management properties to repository entry categories in a versioned or non-versioned repository using the Management Property Organizer, located in the **Repository > Organize Management Properties** menu. These categories include the following:

- Business Object Models
- Query
- Project
- SRL Entities
- Template Entities
- Entry

For details about which category or categories to use for specific project items, see ["Entity Categories" on page 109](#).

You can use a combination of predefined and custom management properties. When you add a management property to category of project item such as a SRL Function or an Instance, you can view the management property information in the Properties tab of an editor in the Blaze Advisor IDE. You can also view management property information for Instances in an RMA. The management property information for decision metaphor instances are displayed in an RMA if they are part of a Group or Code template with no display formatting.

When management properties are added, they are associated with the repository and not a specific project, so that can be used by all the projects in that repository. Each repository you have on your system can have its own set of predefined and custom management properties.

You can build a query using management properties filters to obtain information about specific project items or user activity. See ["Defining a Management Property Expression for Queries and Filters" in *DevelopingRuleProjects.pdf*](#).

A library of management property templates is included with your Blaze Advisor installation. You can modify these management property templates or create your own to create management properties. However because modifying or creating templates is considered an advanced tasks and you can accomplish most of the same tasks, we recommend that you use the Management Properties Organizer in the Blaze Advisor IDE. See ["Creating a Management Property" on page 110](#).

Adding a Management Property to an Entity Category

You can add a management property to an entity category after you create a versioned or non-versioned repository. The project items do not have to exist in a repository for you to associate the management property with the category. As you begin developing your projects you see the management properties you have set for a specific category in the Properties tab of the editors of those project items that belong to that category. For example, if you set a management property such as Creation Date to the SRL Ruleset category. When you create your rulesets in your project, you see the creation date for those rulesets in the Properties tab of the Ruleset Editor. See ["Viewing a Management Property in an Editor" on page 103](#).

You can also use a management property in a query or a filter, see ["Defining a Management Property Expression for Queries and Filters" in *DevelopingRuleProjects.pdf*](#).



Note When adding a management property to a project item category in a versioned repository, you must make sure that the items are *checked out* and *editable*. You can ensure that your project items are checked out and editable, by opening the project in the Project Explorer in the Blaze Advisor IDE and checking out all the items that belong to the category where you will be adding the management property.

To verify that these items are editable, you can open them in an editor, "(read-only)" should not appear on the title bar. If you add management properties to items that are checked out but not editable, errors may be displayed in the Properties tab.

To add a management property to an entity category

- 1 Double-click the project in the Repository Explorer to open it.
- 2 Select **Repository > Organize Management Properties**.
The Management Properties Organizer opens.
- 3 Click the **Categories** tab. See "[Entity Categories](#)" on page 109.
- 4 If you are using a versioned repository, click **Check Out All**.
The **Checkout** icon is displayed for all project items in the **Categories** folder.
- 5 Select a category from the Category Explorer and click **Add**.
In some cases you may need to expand a category to see subcategories.
The Select a Management Property wizard is displayed. See "[Predefined Management Properties](#)" on page 105.
- 6 Select a management property and click **OK**.
- 7 Repeat Steps 5 and 6 to add additional management properties to the same category or another category.
- 8 If you are using a versioned repository, click **Check In Changes**.
A Check In Management Properties window is displayed where you can enter comments.
- 9 If you are using a versioned repository, click **Check In**.
- 10 Click **Close Organizer** to exit from the Management Property Organizer.



Note If you do not check in your changes or cancel your check out, when you attempt to close the Management Properties Organizer, you will be prompted to release the management properties before closing the organizer.

Viewing a Management Property in an Editor

After you add a management property to a project item category, you can view it in the Properties tab of an editor.

To view a management property

- 1 In the Project Explorer, right-click and select **Refresh**.
- 2 In Project Explorer, double-click a project item that belongs to a category where you added a management property.
- 3 Click the **Properties** tab in the editor.

You see the management property information displayed with a default display format. If you want to change the display format. See “[Setting the Advanced Display Options for a Management Property](#)” on page 113.

Editing a Management Property

You can edit a management property in a versioned or non-versioned repository using the Edit Management Property window, which is accessed from the Management Properties Organizer.

You can edit any new management properties that you have not yet checked into the versioning system, the same way you would edit a management property within a repository with no versioning service. However, if you want to edit a management property you have checked in or you want to edit a predefined management property, you need to check it out before you can make any changes.

To edit a management property

- 1 Select **Repository > Organize Management Properties**.

The Management Properties Organizer opens.

- 2 If you are using a versioned repository, click **Check Out All** to edit any management properties that appear in the **All Management Properties** tab.

The Edit button is enabled.

- 3 Click **Edit** in the **All Management Properties** or **Categories** tabs to invoke the Edit Management Properties window.

- 4 Edit the desired fields.

- 5 Make any changes and click **OK**.

- 6 If you are using a versioned repository, click **Check In Changes** and enter any comments.

- 7 If you are using a versioned repository, click **Check In**.

- 8 Click **Close Organizer**.



Note If you do not check in an edited management property, when you attempt to close the Management Properties Organizer, you will be prompted to release the management properties before closing the organizer.

Removing a Management Property

When removing a management property from a project item category in a versioned repository, you must make sure that the items are checked out and editable.

You can verify this by opening the project in the Project Explorer and checking out all the items that belong to the category where you will be adding the management property and open them in an editor. The “read-only” label should not appear in the title bar. If you remove management properties from items that are not editable, errors may be displayed in the Properties tab.

To remove a management property

- 1 Double-click the project in the Repository Explorer to open it.
- 2 Select **Repository > Organize Management Properties**.
The Management Properties Organizer opens.
- 3 Click the Categories tab. See “[Entity Categories](#)” on page 109.
- 4 If you are using a versioned repository, click **Check Out All**.
The Checkout icon is displayed for all project items in the Categories folder.
- 5 Select a category from the Categories Explorer.
The management properties that have been added to that category appear in the editor pane. In some cases, you may need to expand the category to see the subcategories.
- 6 Select a management property and click **Remove**.
- 7 Repeat steps 5 and 6 and remove any additional management properties.
- 8 If you are using a versioned repository, click **Check In Changes**.
A Check In Management Properties wizard is displayed where you can enter comments.
- 9 If you are using a versioned repository, click **Check In**.
- 10 Click **Close Organizer** to exist from the Management Property Organizer.



Note If you do not check in your changes or cancel your check out, when you attempt to close the Management Properties Organizer, you will be prompted to release the management properties before closing the organizer.

Predefined Management Properties

Blaze Advisor provides the following set of predefined management properties for each repository. You can also create your own custom management properties. See “[Creating a Management Property](#)” on page 110.

You can associate a given management property to one or more categories of project items. See “[Entity Categories](#)” on page 109.

When predefined management properties are associated with a given category of project items, you can view the information that the system returns in the Properties tab of an editor. See “[Displaying Management Properties in a Folder or Project Editor](#)” on page 117.

The Type associated with a management property determines the input field you see when you select a management property in a Management Property Filter of a query or a filter.

The following management properties are installed with Blaze Advisor:

Property Name	Description
Creation Date	Type: date Displays the date when the entity was first created.
Current Version	Type: string Displays the repository entry or builtin system attribute values of the same name, using the Repository Entry Information Value Provider. See “ Schema Management of Type Information ” on page 130. For use with a versioned repository. Displays the version number.
Is Versioned	Type: string Displays the repository entry or builtin system attribute values of the same name, using the Repository Entry Information Value Provider. See “ Schema Management of Type Information ” on page 130. For use with a versioned repository. Indicates whether or not the repository is versioned.
Last Modified Date	Type: date Displays the repository entry or builtin system attribute values of the same name, using the Repository Entry Information Value Provider. See “ Schema Management of Type Information ” on page 130. For use with a versioned repository. Displays the date when the entity was last modified.
Last Modified User	Type: string Displays the repository entry or builtin system attribute value of the same name using the Repository Entry Information Value Provider. See “ Schema Management of Type Information ” on page 130. For use with a versioned repository Displays the user identifier which is associated with latest modification of the item.
Predictive Model Type	Type: string String values from a list This management property designates the type of the selected predictive model and is read-only. The possible predictive model types are: <ul style="list-style-type: none">■ Not Applicable■ Neural Network■ Regression Model■ Score Card■ Tree Model

Property Name	Description
Repository Content Type	<p>Type: string</p> <p>Displays the repository entry or builtin system attribute values.</p> <p>The content type of the entry using the Repository Entry Information Value Provider. See “Schema Management of Type Information” on page 130.</p> <p>Displays the content type which can be Fixed (SRL Entity), Instance, Template, or Provider.</p>
Repository Subtype	<p>Type: string</p> <p>Displays the repository entry or builtin system attribute values using the Repository Entry Information Value Provider. See “Schema Management of Type Information” on page 130.</p> <p>The subtype of the entry in the repository.</p> <p>Displays the subtype which can be None, Decision Table, Decision Tree, or Score Model. Most project items have a subtype of None.</p>
Repository Target	<p>Type: string</p> <p>Displays the repository entry or builtin system attribute values using the Repository Entry Information Value Provider. See “Schema Management of Type Information” on page 130.</p> <p>The target of the entry in the repository.</p> <p>Displays the target type as <i>SRL</i> if the item participates directly in the rule project but if the item is used in the repository, the target type is <i>Repository</i>.</p>
Repository Type	<p>Type: string</p> <p>Displays the repository entry or builtin system attribute values of the same name, using the Repository Entry Information Value Provider. See “Schema Management of Type Information” on page 130.</p> <p>The type of the entry in the repository.</p> <p>Displays repository types which can be SRL Entities, Providers, Templates, Instances, Queries, Ruleflows, and Question Sets.</p>
Source Reference	<p>Type: string</p> <p>Displays the repository entry or builtin system attribute values of the same name, using the Repository Entry Information Value Provider. See “Schema Management of Type Information” on page 130.</p> <p>For use with released or published projects.</p> <p>Displays the information on the source directories for released or published projects.</p>

Property Name	Description
Test Case	<p>Type: string</p> <p>This management property is primarily used to designate certain test cases as part of the same test suite in the brUnit Module. By default, this management property is always displayed in a function, ruleset, Function Template instance or Ruleset Template instance.</p> <p>If you make an entry in this field, but do not select a value in the Test Role field, the entity is not used as test case with the brUnit Module.</p> <p>Test Case can be used with other project items, but those items can never be used as test cases and are ignored by the brUnit Module.</p> <p>Note: You see this management property in the Function and Ruleset Templates by default. However, <i>only</i> a value entered in an instance of one of these templates can be used to identify a test case as belonging to a specific test suite. Any Test Case value entered in the Properties tab of a Ruleset or Function Template is ignored by the brUnit Module. See “Unit Testing Rule Projects” in <i>DevelopingRuleProjects.pdf</i>.</p>
Test Role	<p>Type: string</p> <p>String values from a list</p> <p>This management property is primarily used to designate a function, ruleset, a Function Template instance or a Ruleset Template instance as a test case for the brUnit Module.</p> <p>By default, this management property is always displayed in a function, ruleset, Function Template instance or Ruleset Template instance. When you set the value to something other than None, the item is used with the brUnit Module and its contents are not part of the generated SRL.</p> <p>The possible Test Roles you can assign are:</p> <ul style="list-style-type: none"> ■ None ■ Standard Test ■ Set-up ■ Tear-down <p>Test Role can be used with other project items, but those items can never be used as test cases and are ignored by the brUnit Module.</p> <p>Note: You see this management property in the Function and Ruleset Templates by default. However, <i>only</i> a value entered in an instance of one of these templates can be used to designate an entity as a test case. Any Test Role selected in the Properties tab of a Ruleset or Function Template is ignored by the brUnit Module. See “Unit Testing Rule Projects” in <i>DevelopingRuleProjects.pdf</i>.</p>

Property Name	Description
Version Status	Type: string Displays the repository entry or builtin system attribute values of the same name, using the Repository Entry Information Value Provider. See “Schema Management of Type Information” on page 130 . For use with a versioned repository. The status of the entry in a versioned repository. The status can be: <ul style="list-style-type: none">■ Modified■ New■ Up to date.
Working Copy Version	Type: string Displays the repository entry or builtin system attribute values of the same name. See “Schema Management of Type Information” on page 130 . For use with a versioned repository using the Repository Entry Information Value Provider. Displays the version number of the working copy of the item.

The predefined management properties are based on the Management Property Templates in the Management Property Library of the system folder. See [“Management Property Templates” on page 123](#).

Entity Categories

You can add management properties to the following project item categories. The Categories tab of the Management Properties Organizer contains a tree view of the available categories. When you add a predefined or user-defined management property, it is listed in the editor pane of the Categories tab.

The following categories are installed with Blaze Advisor:

Categories	Description
Business Object Model (BOM) <ul style="list-style-type: none">■ COBOL BOM■ Custom BOM■ .NET BOM■ Java BOM■ Database BOM■ XML BOM	Used to add management properties to a business object model.
Entry	Used to add management properties to all projects and project items, except folders. You cannot set management properties for folders.
Project	Used to add management properties to projects.
Query	Used to add management properties to queries and verification instances.

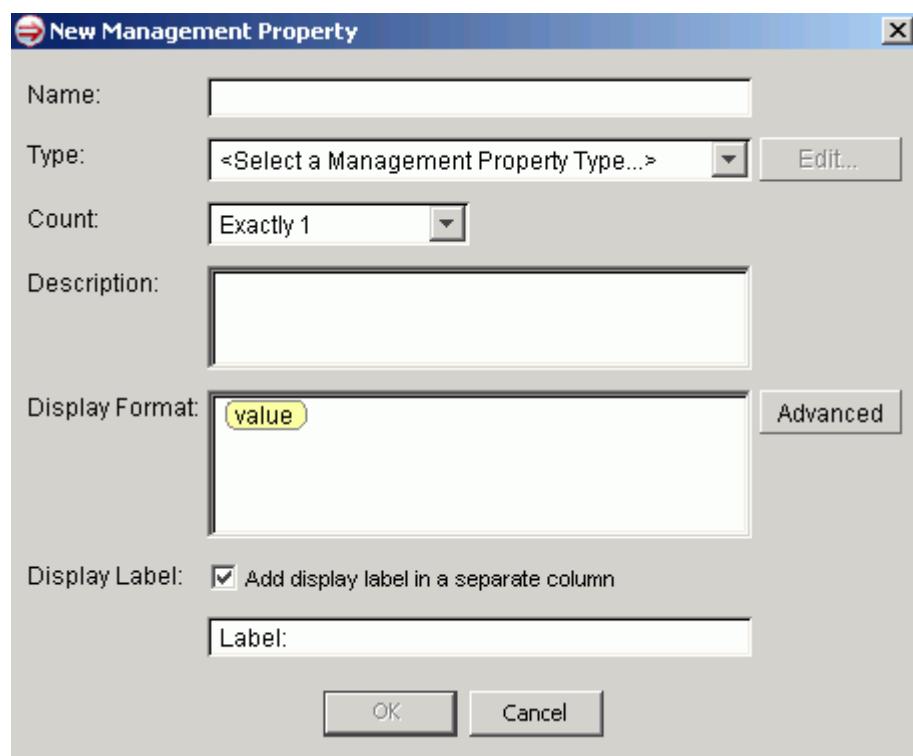
Categories	Description
SRL Entities <ul style="list-style-type: none">■ Decision Table■ Decision Tree■ Question Set■ Ruleflow■ Score Model■ SRL Class■ SRL Enumeration■ SRL Event Rule■ SRL Function■ SRL Named Object■ SRL Pattern■ SRL Reason Code List■ SRL Rule■ SRL Ruleflow Functional■ SRL Ruleset■ SRL Variable	Used to add management properties to specific SRL entities.
Template Entities <ul style="list-style-type: none">■ Instance■ Provider■ Template	Used to add management properties to specific Template entities. You can view management property information for most instances in an RMA. The management property information for decision metaphor instances are displayed in an RMA if they are part of a Group or Code template with no display formatting. If you want to set management properties for filters, use the Instance category. Note that any management properties set for the Instance category also appear in queries and verification instances.

The entity categories are based on the Management Property Category Templates in the Management Property Library of the system folder. See ["Management Property Templates" on page 123](#).

Creating a Management Property

You can create a new management property using the New Management Property window in a versioned or non-versioned repository. You access this window from the Management Properties Organizer in the Blaze Advisor IDE. If you are working with a versioned repository, when you have created a new management property, you need to check in your changes. Once you create the management property, you can add it to any project item category.

This is how the New Management Property window looks when you open it in the Blaze Advisor IDE:



In this window, you can do the following:

- Name the management property
 - Select the type that you want to use. There are several available types. See the Type column in ["Advanced Display Options" on page 114](#).
- When you want to create a management property that uses a value list. See ["Creating a Management Property Value List" on page 116](#).
- Select the count or number of fields that you want displayed. For example, if you are using an "Author" management property, you may want to allow users to display more than one.
 - Describe the management property you are creating. This description appears in the Description columns in the All Management Properties tab.
 - Set the Display Format which affects the display of the management property value in the Properties tab of an editor.
 - (Optional) You can also use the Advanced button to specify how the management property information is displayed. See ["Setting the Advanced Display Options for a Management Property" on page 113](#).
 - Set the Display Label which affects the display of the management property field in the Properties tab. By default, Blaze Advisor aligns all management property fields on the left-hand side of the tab, as shown below:

Active:	<input type="text" value="true"/>
Author:	<input type="text"/>
Creation Date:	<input type="text" value="Mar 8, 2005"/> 
Repository Content Type:	

Instead of using the default behavior, you can clear the **Add display label in a separate column** option to use free-form labels or HTML to display the management property fields in the Properties tab of an editor.

When you create a management property, you do not have to explicitly save it. After you click the OK button, the management property is registered in the Select a Management Property window and it is ready to be added to a project item category. See “[Adding a Management Property to an Entity Category](#)” on page 102.

To create a new management property

- 1 Select **Repository > Organize Management Properties**.

The Management Properties Organizer opens.

- 2 Click the **All Management Properties** tab, if it is not already displayed.
- 3 (Optional) If you are using a versioned repository, click **Check Out All**.
- 4 Click **New** to invoke the New Management Property window.
- 5 Enter a name for the management property.
- 6 Select the type of the management property from the **Type** drop-down list.
- 7 Select the number of times the field can be added in the Management Properties tab from the **Count** drop-down.
- 8 (Optional) Enter text that describes what the management property is used for in the **Description** window. This description will appear in the All Management Properties tab of the Management Properties Organizer.
- 9 (Optional) Enter any HTML that you want used to display the value field in the **Display Format** window.
- 10 (Optional) Click **Advanced** to specify how the management property is displayed in the field.

The fields you see in the Advanced Display Options wizard, depend on the type of the management property you are creating. See “[Setting the Advanced Display Options for a Management Property](#)” on page 113.

11 (Optional) Clear the check box next to **Add display label in a separate column**.

If you clear this option, the field for editing the label disappears. You can add HTML in the Display Format window to format the display for a property label and specify how the property information is displayed.

Alternatively, if you retain this option, you can edit the field by highlighting the existing text and entering a label name.

12 Click **OK** or **Cancel**.

If OK is not enabled, it is because you did not enter a name in Step 4 or you did not select a type Step 5.

13 Click **Check In Changes**.

A Check In Management Properties wizard is displayed where you can enter comments.

14 Click **Check In**.

Alternatively, click Cancel to exit the Check In Management Properties dialog.

Note If you do not check in or cancel the check out, when you attempt to close the Management Properties Organizer, you will be prompted to release the management properties before closing the organizer.

15 Click **Close Organizer**.

Once you create the management property you can add it to any entity category. See "[Adding a Management Property to an Entity Category](#)" on page 102.

Setting the Advanced Display Options for a Management Property

You can set advanced display options for an existing management property. The display options that are available depend on the management property type.

To set advanced display options

- 1 Open the Management Properties Organizer.
- 2 If you are using a versioned repository, click the **Check Out All** button.
- 3 On the **All Management Properties** tab:
 - If you want to set display options for a new management property, click **New**.
 - If you are editing the display options for an existing management property, select the management property, and click **Edit**.

Depending on the management property type, you see different options when you click **Advanced**. See "[Advanced Display Options](#)" on page 114.

- a Select and enter values for the display options.
- b Click **OK** to exit from the Advanced Display Options window.

- 4 If you are using a versioned repository, click **Check In All** and enter comments in the Check In Management Properties window and click **Check In**.
- 5 Click **Close Organizer**.

Advanced Display Options

The Advanced Display Options for new or existing management properties are described in the following table:

Type	Options	Description
boolean	If the Count is Exactly 1, the following options are available: Selection method: <ul style="list-style-type: none">■ Radio Buttons■ Check box (default)■ Javascript/DHTML Choicebox■ Javascript/DHTML Mini Choicebox If the Count is something other than Exactly 1, the following options are available: Selection method: <ul style="list-style-type: none">■ Choicebox (default)■ Javascript/DHTML Choicebox■ Javascript/DHTML Mini Choicebox Page Location <ul style="list-style-type: none">■ Same Page■ New Page If the Count is: <ul style="list-style-type: none">■ User-Defined■ User-Defined-No Duplicates you have the same options as if the Count is other than Exactly 1 but you can also define a Min occurrence and a Max occurrence.	If you choose to user a User-Defined or User-Defined-No Duplicates, you can also define a Minimum (Min) and a Maximum (Max) count on the New Management Property window or the Edit Management Property window.

Type	Options	Description
current version date integer is versioned last modified date last modified user repository content type repository subtype repository target repository type version status working copy version	If the Count is Exactly 1, the following options are available: Field Size Style <ul style="list-style-type: none">■ Standard■ Dhtml If the Count is something other than Exactly 1, the following options are available: Field Size Style <ul style="list-style-type: none">■ Standard■ Dhtml Page Location <ul style="list-style-type: none">■ New Page■ Same page If the Count is User-Defined you have the same options as if the Count is something other than Exactly 1 but you can also define a Min occurrence and a Max occurrence.	The Field Size is for text input field that is displayed in the Properties tab. You enter a width for the text input field using a positive integer such as 3, 10, 21, and 30. If this field is left blank, Blaze Advisor generates an input field using a default width. If the Count is User-Defined, you can also define a Minimum (Min) and a Maximum (Max) count on the New Management Property window or the Edit Management Property window.
string	If the Count is Exactly 1, the following options are available: Field Size Style <ul style="list-style-type: none">■ Standard■ Dhtml Key Echo <ul style="list-style-type: none">■ Normal Echo■ Password Style If the Count is other than Exactly 1, the following options are available: Field Size Style <ul style="list-style-type: none">■ Standard■ Dhtml Key Echo <ul style="list-style-type: none">■ Normal Echo■ Password Style Page Location <ul style="list-style-type: none">■ New Page■ Same page If the Count is User-Defined, you have the same options as if the Count is other than Exactly 1 but you can also define a Min occurrence and a Max occurrence.	The Field Size is for text input field that is displayed in the Properties tab. You enter a width for the text input field using a positive integer such as 3, 10, 21, and 30. If this field is left blank, Innovator generates an input field using a default width. The Style is the style of the text box. <ul style="list-style-type: none">■ Standard is an inset text box.■ Dhtml is a linear text box. The Key Echo is how the user input is displayed. <ul style="list-style-type: none">■ Normal Echo displays user input exactly as it is entered.■ Password Style hides user input using a set of "****". If you choose to user a User-Defined, you can also define a Minimum (Min) and a Maximum (Max) count on the New Management Property window or the Edit Management Property window.

Type	Options	Description
Test Roles Predictive Model Type	If the Count is Exactly 1, the following options are available: Selection method: <ul style="list-style-type: none"> ■ Radio Buttons ■ Check box (default) ■ Javascript/DHTML Choicebox ■ Javascript/DHTML Mini Choicebox If the Count is something other than Exactly 1, the following options are available: Selection method: <ul style="list-style-type: none"> ■ Choicebox (default) ■ Javascript/DHTML Choicebox ■ Javascript/DHTML Mini Choicebox Page Location <ul style="list-style-type: none"> ■ Same Page ■ New Page If the Count is: <ul style="list-style-type: none"> ■ User-Defined ■ User-Defined-No Duplicates, you have the same options as if the Count is other than Exactly 1 but you can also define a Min occurrence and a Max occurrence.	
New Value List	Launches the New Value List wizard.	See “ Creating a Management Property Value List ” on page 116.

Creating a Management Property Value List

You can create a drop-down list for a management property by using the Management Properties Organizer.

To create a management property value list

- 1 If you have not already done so, invoke the Management Properties Organizer and click **New**.
- 2 Click **New** to create a new management property.
- 3 Select **New Value List** from the **Type** drop-down list.
The New Value List wizard opens.
- 4 Enter a name for the value list in the **Name** field.
- 5 Enter a string value in the **New Value** field and click **Add**.
- 6 (Optional) To remove a string value from the value list, select the string value from **Values** and click **Remove**.
- 7 When you have completed your list, click **OK** to exit the wizard.

The value list you created appears in the Type drop-down list in the New Management Property window.

Displaying Management Properties in a Folder or Project Editor

When management properties are applied to a category of project items, they are displayed on the Properties tab of an editor in the Blaze Advisor IDE. If the management properties are applied to the Instance category, they are displayed in an RMA. The management property information for decision metaphor instances are displayed in an RMA if they are part of a Group or Code template with no display formatting.

You can also display management properties associated with project items in the Content tab of a Project and Folder Editor. The Project Editor displays the management properties in a list and the Folder Editor displays the management properties in a table.

The management properties you add in the Folder or Project Editor, also appear as columns in the Results table of a query.



Note You will not be able to display management properties in the Project or Folder editors until you have added them to the project items category. See "[Adding a Management Property to an Entity Category](#)" on page 102.

To display management properties in a Folder or Project Editor

1 To display the project editor or the folder editor, follow one of the following instructions:

- In the Project Explorer, select the project file and right-click and select the **Open Project Editor** command.
- In the Project Explorer, select the folder and right-click and select the **Open Folder Editor** command.

The Project or Folder Editor is displayed in the Editor pane.

2 If you are using a versioned repository, you need to check out the Project or Folder before you can add a management property for display.

3 Click **Select columns** icon from the **Project** or **Folder Editor** toolbar or the Contents tab toolbar.

The **View Folder Options** dialog window is displayed.

4 Click **Add**.

The Select a Management Property window is displayed.

5 Click on the management property you want to display.

6 Click **OK**.

7 Repeat Steps 3 to 5 to display other management properties.

The management properties appear in the order they were added to the View Folder Options dialog window.

8 (Optional) To change the display order of the management properties you can use the **Move Up** or **Move Down** buttons.

- 9 If you are using a versioned repository, you need to check in your changes either by using the **Check In** command in the context menu or the **Check In** icon on the toolbar.

Any changes to the order of management properties you make does not affect how the management properties appear in the Properties tab.

Removing Management Properties from a Project or Folder Editor

When you remove management properties from the Content tab of the Project or Folder Editor, it does not affect how the management properties display in the Properties tab of a project item editor.

To remove a management property from the Content tab

- 1 To display the project editor or the folder editor, follow one of the following instructions:
 - In the Project Explorer, select the project file and right-click and select the **Open Project Editor** command.
 - In the Project Explorer, select the folder and right-click and select the **Open Folder Editor** command.
- The Project or Folder Editor is displayed in the Editor pane.
- 2 If you are using a versioned repository, you need to check out the Project or Folder before you can add a management property for display.
- 3 Click **Select Columns** icon  from the **Project or Folder Editor** toolbar or the Contents tab toolbar.
The View Folder Options dialog window is displayed.
- 4 Select a management property you want to remove from the editor and click **Remove**.
- 5 Click **OK**.
- 6 Repeat Steps 3 and 4 to remove any additional management properties.
- 7 If you are using a versioned repository, you need to check in your changes either by using the **Check In** command in the context menu or the **Check In** icon on the toolbar.

Cutting, Copying, and Pasting Project Items with Management Properties

If you want to copy and paste project items with management properties to another repository, for best results, you need to take some steps to prepare the target repository where you want to paste your project items. See "Cutting, Copying, and Pasting Project Items" in *DevelopingRuleProjects.pdf*.

If you want to retain the same set of management properties, you need to make sure that the repository where you will be pasting your project items into, has the same configuration and the same management properties.

For example, if you are working in a shared workspace using a File BVS repository connection that has management properties for versioning information, BEFORE you begin copying and pasting project items, you need to configure the repository where you will be pasting your project items to use file BVS versioning with a shared workspace and add the same set of management properties.



Important The version history information cannot be copied and pasted into the new repository.

If you want to use a different set of management properties, you need to make sure that the repository where you will be pasting your project items has the repository configuration and the management properties you want to use. These management properties will be used instead of the management properties of the source repository.

Managing Templates and Instances in Your system Folder

You can manage the templates and instances of queries, filters, verifications, and management properties in the system folder. This section includes the following topics:

- “Displaying the system Folder” on page 119
- “system Folder Contents” on page 120
- “Management Property Templates” on page 123
- “Management Property Types” on page 128
- “Schema Management of Type Information” on page 130

Displaying the system Folder

A *system folder* is included in all repositories you create in Blaze Advisor. When you create a repository, a copy of the system folder located in <ADVISOR_HOME>\lib\Admin Repository is added to each repository you create in Blaze Advisor. The system folder contains several projects and folders that provide the ability to run queries based on user-defined criteria in the Blaze Advisor IDE and in rule maintenance applications, filter project items, verify rule projects, specify and create management properties, and support business rule templates. The support for business rule templates are currently only for backwards compatibility purposes.

By default, the system folder in each repository is hidden, unless you choose to display it. You can only view the system folder for a repository in the Repository Explorer. Because the system folder can be shown or hidden and is never exposed to business users, it can be used to store files and folders that you want to make available to a select group of users.

If you created repositories using an earlier version of Blaze Advisor and you install a newer version of Blaze Advisor, you may not be able to take advantage of the latest versions of the system templates unless you update the system templates in the system folder. See “[Synchronizing the system Folder](#)” on page 137.

To view the contents of the system folder

- 1 If you have not already done so, select **Windows > Preferences > Blaze Advisor** and select the **Project and Object Model Explorer Views**.
- 2 Select **Show system libraries and folders** option and click **OK**.
- 3 Select **Windows > Preferences > Blaze Advisor** and select the **Project and Object Model Explorer Views** and select **Show advanced commands for creating and managing repositories**.
- 4 Select **Repository > Local Admin Repository**.

The Connection and Configuration folders of the Admin Repository and the Business Terms Support Library Folder, Custom Table Filter Folder, Management Properties Library Folder, and Queries Library Folder in the system folder are displayed in the Project Explorer.



Important If you have installed a new version of Blaze Advisor and you want to use project in your existing 6.0 repositories with the newer version of Blaze Advisor, you need to update the system templates of these repositories by running a command-line utility. See “[Synchronizing the system Folder](#)” on page 137.

system Folder Contents

The system folder contains the support library projects and folders to allow users to specify and run queries, use and specify management properties, and support business rule templates (backwards compatibility for earlier releases.) The three main libraries that will be discussed in this section are the Query Library and the Management Property Library. You can create and edit the templates from these libraries and use the changes in the Blaze Advisor IDE and the rule maintenance application if the predefined queries and management properties provided in the Blaze Advisor IDE do not provide you with the types of information you need about your project or project items.

This section contains the following topics:

- [“Management Properties Library” on page 120](#)
- [“Query Library” on page 121](#)



Note The Business Terms Support Library is included in the system folder for backward compatibility purposes only.

Management Properties Library

Management properties provide a way to extend the amount of data available for a entities in a given project. Blaze Advisor provides a set of predefined management properties. These management properties are defined by the templates and providers

you see in the Management Properties Library subfolders. You can apply these management properties to a repository by using the Management Property Organizer in Blaze Advisor. You can invoke this wizard by choosing the Organize Management Properties command from the Repository menu. After you apply management properties in a repository, any entities you create that belong to the categories with a management property will display the property information on the Properties tab in the editor.

You can also create a custom Management Property Template in the system folder of the Admin Repository or Blaze Advisor. Both predefined and custom templates can be added to an entity category using the Management Property Organizer.

The Management Properties Library project contains the following folders:

- **Management Property Category Templates**

The Management Property Category Templates folder contains templates that define the categories of entity types you see in the tree view of the Categories tab of the Management Property Organizer. The categories include Business Object Models such as the Java BOM, XML BOM, COBOL BOM, and Database BOM, SRL Entities such as rulesets, rules, functions, and classes, and Template Entities such as templates, instances, and providers. Other categories include Entry, Project and Query. The Entry category is used when you want to apply a management property to all category types.

- **Management Property Templates**

The Management Property Templates folder contains the set of predefined management properties you see All Management Properties tab of the Management Properties Organizer. You can apply to a property to a category such as a Entry, Project, an SRL Entity such as a SRL Ruleset, or a Template Entity such as an Instance.

The management property templates define the display format and the value type for a given management property. For example, the Last Modified Date management property has a text field that allows a date value to be entered.

- **Management Property Types**

The Management Property Types folder contain providers that define the value type allowable for the given management property defined by one of the management property templates. The Type drop-down list in the New Management Property window of the Management Properties Organizer in the Blaze Advisor IDE displays the value types in this folder. An example of a type would be boolean, string, or date.

Query Library

A query allows you to find all the entity files that match a given search term(s) within the scope of a project based on a set of criteria you define. There are predefined query templates in the Query Library subfolder that are examples of the types of customized queries you can create using a Query Template.

You can also use the Standard Query or Standard Business Query from the Blaze Advisor IDE or a rule maintenance application and build and save a query instance by using the provided fields. The commands for the predefined queries appear in the Search submenu located in the Edit menu. If you create a custom query and save it to the Query Library, projects in your repository can select your query template from the same Search submenu in the Blaze Advisor IDE or in a rule maintenance application.

The Query Library contains three folders, each containing a predefined query:

- **Builder Filters**

The Builder Filters folder contains a Standard Filter folder. The Standard Filter folder contains the filter templates used in the fields of the Standard Filter Instance Editor in the Blaze Advisor IDE. When you open the Filter templates in a editor you see that they contain XML in the XML Content section for the filter configuration. To use a filter, see "Filtering Project Content" in *DevelopingRuleProjects.pdf*.

Builder Queries

The Builder Queries folder contains a Standard Query template that defines how the fields in a Standard Query are displayed and what values may be entered in the query fields. The template also contains SRL in the function body to provide instructions on how the search mechanism uses the search criteria defined by the user.

When the Standard Query instance is invoked, you can view it in the Query Editor in the Blaze Advisor IDE or in a rule maintenance application. The Standard Query is used by rule developers and project managers to find information on files in a project. See "Defining a Standard Query" in *DevelopingRuleProjects.pdf*.

- **Business Queries**

The Business Queries folder contains the Standard Business Query template that defines how the fields should be displayed and what values may be entered in the query fields. The template also contains SRL in the function body to provide instructions on how the search mechanism uses the search criteria defined by the user.

When a Standard Business Query instance is invoked, you can view it in the Query Editor in the Blaze Advisor IDE or in a rule maintenance application. The Standard Business Query is designed to be used primarily by business users who want to search instance files for display content. See "Creating and Running Queries in an RMA" in *DevelopingRuleMaintenanceApplications.pdf*.

- **Verification Queries**

The Verification Queries folder contains the Verification Query template that defines how the fields of the Verification instance are displayed in the Verification Editor in the Blaze Advisor IDE. The template also contains SRL in the function body to provide the search mechanism using the criteria defined by the user.

Management Property Templates

You can create and edit management property templates in the system folder of your Admin Repository. If you create a new management property in the Admin Repository, any new repositories you create will have access to the new property and the new property also appears in the Management Properties Organizer in the Blaze Advisor IDE. If you are working with more than one repository and you want to use a given management property for projects in each repository, you need to add it to each repository.

Because creating a new Management Property Template in the system folder is considered an advanced task, we recommend that you use the Management Properties Organizer when you want to create, delete, and edit a new Management Property for a given repository. See ["Creating a Management Property" on page 110](#).

About Management Property Templates

Management Property Templates define management properties that can be assigned to a category of entities within a repository. The Management Property Templates include a property type, defined by a value holder referring to a provider, and a display format for how the template appears in the Property tab when the contents for the entity that uses the property is displayed in the Blaze Advisor IDE. For information about the providers used by these templates, see "Provider Reference" in *DevelopingRuleMaintenanceApplications.pdf*.

For information about the Repository Entry Information Value Provider, see ["Management Property Providers" on page 129](#).

The following table describes the predefined management property templates provided in your Blaze Advisor installation:

Property Name	Description
Creation Date	The date when the entity was first created. Treats the date as a string value. Uses a Date Provider.
Current Version	Used for versioned repositories. The current version of the entry in a repository with versioning. Uses a Repository Entry Information Value Provider.
Is Versioned	Used for versioned repositories. Indicates whether or not the entity is versioned. The default is true. Uses a Repository Entry Information Value Provider.
Last Modified Date	Used for versioned repositories. The date when the entity was last modified. Uses a Repository Entry Information Value Provider.
Last Modified User	Used for versioned repositories. The user name associated with latest modification of the entity. Uses a Repository Entry Information Value Provider.

Property Name	Description
Predictive Model Type	This management property indicates the type of a predictive model and is read-only. In the Properties tab for the model's invocation method (which has the same name as the model), users can view the predictive model type. Uses a Localized String List Provider. The possible predictive model types are: <ul style="list-style-type: none">■ Not Applicable■ NeuralNetwork■ RegressionModel■ ScoreCard■ TreeModel
Repository Content Type	The content type of the entry in the repository managed by the schema. Uses a Repository Entry Information Value Provider.
Repository Subtype	The subtype of the entry in the repository managed by the schema. Uses a Repository Entry Information Value Provider.
Repository Target	The target of the entry in the repository managed by the schema. Uses a Repository Entry Information Value Provider.
Repository Type	The type of the entry in the repository managed by the schema. Uses a Repository Entry Information Value Provider.
Source Reference	The source information for entities of release or published projects.
Test Case	The test case identifier. By default, this management property is displayed for SRL Rulesets and Functions. In the Properties tab of the Ruleset or Function editor, users can enter a string value for the Test Case. Uses a String Provider.
Test Role	The test role status. By default, this management property is displayed for SRL Rulesets and Functions. In the Properties tab of the Ruleset or the Function editor, users can select a test role string value from a drop-down list. Uses a Localized String List Provider. The Test Roles include: <ul style="list-style-type: none">■ None■ Standard Test■ Set-up■ Tear-down
Version Status	The status of the entry in a versioned repository. Uses a string value. Uses a Repository Entry Information Value Provider.
Working Copy Version	The current version of the working copy of the entry in a versioned repository. Uses a Repository Entry Information Value Provider.

Creating a New Management Property Template

You can create new management property templates and add them to the Management Property Template folder in the system folder of your Admin Repository. To be displayed correctly in the Management Property Organizer, the value holder name must be *value* and its management property type must be *string*.

In the Blaze Advisor IDE, you can use the Management Properties Organizer to create a new management property. See ["Creating a Management Property" on page 110](#).

To add a new Management Property in the Admin Repository

- 1 If you have not already done so, connect to the Admin Repository. See “[Connecting to the Local Admin Repository](#)” on page 132.
- 2 If you have not already done so, display the system folder, see “[Displaying the system Folder](#)” on page 119.
- 3 Expand the system folder in Repository Explorer.
- 4 Double-click the **Management Properties Library** project.
The Management Properties Library contents are displayed in Project Explorer.
- 5 Right-click Management Property Templates and select **New > Template Entities > Template**.
The New Template window is displayed.
- 6 Select the **Rule Management** tab and select **Management Property**.
- 7 Click **OK**.
The Management Property Template Editor opens.
- 8 Define the Management Property Template by entering values in the editor.
 - a In the **Management Property Template** field, highlight the default text and enter a name for the management property template.
 - b (Optional) Enter a name in the **Template Display Name** field.
 - c Click **New Value Holder** icon  to add a value holder.
 - Enter the text “*value*” for the name of the value holder.
 - In the **Type** drop-down select **Select from Project**.
The Select from Project window opens.
 - Expand the Management Properties Library Folder > Management Property Types folder and select the **string** management property type and click **OK**.
 - Select a count from the **Count** drop-down.
 - (Optional) Enter a name in the **Display Name** field.
 - d Enter the display format you want to use in the **Display Format** field.
 - e Click in the **Display Format** field and click the **Insert Placeholder** icon  to add a display placeholder referring to the template you are creating.
 - f Enter HTML to format how the management property appears in the Properties tab when the contents of a given entity are displayed in an editor.
- 9 Click **Save All**.

Editing a Management Property Template

You can edit a management property template if you want to change some of its values, however if a category of entities is currently using the management properties, you may cause errors to occur. In the Blaze Advisor IDE, you can also use the Management Properties Organizer to edit a management property. See ["Editing a Management Property" on page 104](#).

To edit a Management Property in the Local Admin Repository

- 1 If you have not already done so, connect to the Local Admin Repository. See ["Connecting to the Local Admin Repository" on page 132](#).
- 2 If you have not already done so, display the system folder, see ["Displaying the system Folder" on page 119](#).
- 3 Expand the system folder in Repository Explorer.
- 4 Double-click the Management Properties Library project.
The Management Properties Library contents are displayed in Project Explorer.
- 5 Double-click a management property to open its contents in the editor.
- 6 Edit the relevant fields.
- 7 Save your changes.

The changes you have made will appear in the Management Properties Organizer in the Blaze Advisor IDE and in each new repository you create

Deleting a Management Property Template

In the Blaze Advisor IDE, you can remove a management property so that it is no longer used to provide information for a category of project entity, but it is still available for other categories of entities. See ["Removing a Management Property" on page 105](#).

When you delete a management property template in the system folder, the action cannot be reverted. You will no longer be able to use the property in the Management Properties Organizer in the Blaze Advisor IDE.

To delete a Management Property Template from the Admin Repository

- 1 Before you delete the management property template, you need to make sure that there are no entity categories that still use the property.
You can use the Management Property Organizer in the Blaze Advisor IDE and remove the management property from the categories where it has been applied. This is the recommended approach. See ["Removing a Management Property" on page 105](#).
- 2 After you have removed the management property from the categories where it was applied, you need to connect to the Admin Repository. See ["Connecting to the Local Admin Repository" on page 132](#).

- 3 Display the system folder, see “[Displaying the system Folder](#)” on page 119.
 - a Expand the system folder in Repository Explorer.
 - b Double-click the **Management Properties Library** project.
 - c The Management Properties Library contents are displayed in Project Explorer.
 - d Expand the **Management Property Category** folder.
 - e Select the category or subcategories that use the management property that you want to delete.
 - f Open each category template in the editor pane to remove the value holder and placeholder referring to the management property template.
- 4 After you have completely removed all references to the property template, choose the property template and use the Delete command from the right-click menu. You receive a message informing you that if you proceed with the deletion that references to the template will be invalid.
- 5 Click **Yes**.
The property template is removed from the folder.
- 6 Save your changes.

Adding a Management Property to a Category in the system Folder

You can add a management property to a category of entity by adding a Management Property Template to a Management Properties Category Template in the Management Properties Library. However, in most cases, we recommend that you use the Management Properties Organizer to add management properties to a category of entities. See “[Adding a Management Property to an Entity Category](#)” on page 102.

To add a management property to a category of entity in the system folder

- 1 From the Management Property Category Templates folder, choose a template by double-clicking it to view its contents in an editor.
- 2 Click **New Management Property** icon  from the **Template Editor** toolbar or to the right of the **Value Holders and Arguments** section.
- 3 Enter a name for the Management Property.
- 4 Select a Template or Provider from the **Type** drop-down.
- 5 Select a count from the **Count** drop-down.



Note To avoid errors, the count of the management property Value Holder being added to a category of entities should match the count used for the Value Holder of the Management Property Template. In most cases, the count for the management should be set to Exactly 1.

- 6 (Optional) Enter a name for the **Display Name**.
- 7 (Optional) Enter a description in the **Description** field.

Management Property Types

You can create and edit management property type providers in the system folder of your Admin Repository. If you create a new management property type provider in the Admin Repository, any new repositories you create will have access to the new provider and the new provider also appears in a drop-down list in the New Management Property window of the Management Properties Organizer in the Blaze Advisor IDE.

Because creating new management property type providers in the system folder is considered an advanced task, in general we recommend that you use the Management Properties Organizer when you want to create a new value list for a set of text strings. See [“Creating a Management Property” on page 110](#).

About the Management Property Types

Blaze Advisor provides a set of providers that can be used to define the allowable type for a given management property. The following table describes the property types providers that are installed with Blaze Advisor.

For information about the providers used, see [“Provider Reference” in *DevelopingRuleMaintenanceApplications.pdf*](#).

For information about the Repository Entry Information Value Provider, see [“Management Property Providers” on page 129](#).

Property Type	Created Using the Following Provider	Used in These Predefined Management Property Templates
boolean	Boolean Provider	
current version	Repository Entry Information Value Provider	Current Version
string	String Provider	
date	Date Provider	Creation Date
integer	Integer Provider	
is versioned (string)	Repository Entry Information Value Provider	Is Versioned
last modified date (date)	Repository Entry Information Value Provider	Last Modified Date
last modified user (string)	Repository Entry Information Value Provider	Last Modified User
Predictive Model Types (string)	Localized String List Provider	Predictive Model Type

Property Type	Created Using the Following Provider	Used in These Predefined Management Property Templates
repository content type (string)	Repository Entry Information Value Provider	Repository Content Type
repository subtype (string)	Repository Entry Information Value Provider	Repository Subtype
repository target (string)	Repository Entry Information Value Provider	Repository Target
repository type (string)	Repository Entry Information Value Provider	Repository Type
Test Roles (string)	Localized String List Provider com.blazesoft.template.engine.provider.NdLocalizedStringProvider	Test Role
version status (string)	Repository Entry Information Value Provider	Version Status
working copy version (string)	Repository Entry Information Value Provider	Working Copy Version
New Value List	Value List Provider	You can use this property type to create a drop-down list of string values for a new management property you are creating. See “Creating a Management Property Value List” on page 116 .

Management Property Providers

Blaze Advisor provides several specific providers for use with management properties. These providers can be added when you go to the **Utility** tab of the New Provider window. You can open the New Provider window by selecting **File > New Template Entities > Provider**.

choosing New Provider from the Templates and Providers submenu located under the Project menu.

- Management Property Names

Used when building rule maintenance applications. You create a Management Property Names provider when you want to allow the user to obtain the fully qualified names of all of the properties that apply for a given category or categories of entities.
- Management Property Type

Used when building rule maintenance applications. You create a Management Property Type provider when you want the user to obtain the SRL type for a given category of entities.

- Management Property Values

Used when building rule maintenance applications. You create a Management Property Values provider when you want to obtain the current value for a property of a given category of entities.

- Repository Entry Information Value

The Repository Entry Information Value provider is used to obtain the value of any repository entry attribute or system built-in attribute. These string attributes are stored and updated by Blaze Advisor in the `.innovator_attbs` file for each item.



Note Repository entry attributes are only generated when a repository item is saved for the first time. Therefore, the entry attribute will not be seen until the item is saved.

The property types using the Repository Entry Information Value provider are:

- repository target
- repository type
- repository subtype
- repository content type

The values for these property types are stored in the `.innovator_attbs` files in the storage system. The data type for these values is string.

See ["Schema Management of Type Information" on page 130](#).

These property types are updated and retrieved when an entry is saved and checked into a repository with versioning.

- current version
- is versioned
- last modified date
- last modified user
- version status
- working copy version

These property types allow users to enter or choose a value from a drop-down list and use it in a filter.

- test roles
- test cases

This property type retrieves data from a released or published entry.

- source reference

Schema Management of Type Information

Although the repository contains typed content, such as SRL rulesets, ruleflows, or deployment, it does not contain type information. The type information of the content is

managed using a schema manager. Each repository configuration contains a default schema manager. Each entity type is described by a schema element. You can obtain meta-information from the schema element. The meta-information of a type contains four attributes: *type*, *sub-type*, *content type*, and *target*. These attributes are stored and updated in the *.innovator_attbs* file for each item. These attributes can be obtained for a category of entity by using management properties. You can view the information in the Properties tab of an editor. See “[Viewing a Management Property in an Editor](#)” on page 103.

Type Attribute - Repository Type

The type describes the type of the entity and includes:

- Any SRL Entities including SRL Rule, SRL Ruleset, SRL Pattern, and SRL Class.
- Any Template and Instance types including Group, SRL Project Items, and SRL Ruleset Items.
Decision Table, Decision Tree, and Score Model Templates have a type of SRL Ruleset.
- Any Provider types including: Money, Database, and Boolean.
- Any XML type including RMA Definition, Rule Service Definition, Deployment Definition, and System Definition.
- Other entities have a type of Filter, Question Set, Ruleflow, and Query.
The Query, Comparison and Verification are of Query type.

Subtype Attribute - Repository SubType

The subtype describes the entity’s categorization within a type as a string value. For example, an Decision Table entity is a subtype of the SRL Ruleset type and a Comparison is a subtype of the Query type.

Content Type Attribute - Repository Content Type

The content type describes the entity’s content as a string value. The content type can be: Fixed (SRL Entity), an Instance, a Template or a Provider.

Target Attribute - Repository Target

A target describes what the entity is used for as a string value. There are two target types: SRL or Repository. The Decision Table Template, is an example of an entity that has a target of SRL. The RMA Definition is an example of an entity that has a target of Repository.

Managing Repository Configuration and Connection Instances in the Admin Repository

You can manage the configuration and connection templates and instances in the Admin Repository that is installed with Blaze Advisor.

This topic contains the following sections:

- “Admin Repository” on page 132
- “Connecting to the Local Admin Repository” on page 132
- “Contents of the Local Admin Repository” on page 133
- “Editing Configurations and Connections Instances” on page 134
- “Editing Connection Templates” on page 135
- “Editing Connection Instance Files” on page 136

Admin Repository

The Admin Repository is a file repository located in the <ADVISOR_HOME>/lib directory of your Blaze Advisor installation and is not shared with other users. This repository is hidden by default. The Admin Repository contains the underlying templates and instances that control the repository types and repository connections available in your installation of Blaze Advisor. See “[Connecting to the Local Admin Repository](#)” on page 132.

The Admin Repository also contains the system folder that contains the templates and instances for the management features in Blaze Advisor. See “[Managing Templates and Instances in Your system Folder](#)” on page 119.

Editing of templates and instance files in the Admin Repository is considered an advanced task and we recommend that only advanced Blaze Advisor users or administrators edit or delete files in the Admin Repository.



Note Changes made in the Admin Repository may affect how you use the Blaze Advisor. It is therefore recommended that you make a backup copy of the Admin Repository folder before making any changes. The Admin Repository folder is in the <ADVISOR_HOME>/lib folder of your Blaze Advisor installation.

Connecting to the Local Admin Repository

You can connect to the Admin Repository by selecting an option that displays the Local Admin Repository command in the Tools > Repository Administration menu.

To connect to the Admin Repository

- 1 Select **Window > Preferences > Blaze Advisor > General**.
- 2 On the General Options page, select the **Show advanced commands for creating and managing repositories** option and click **OK**.
- 3 Select **Repository > Local Admin Repository**.

The Project Explorer opens displaying a Connection project that references the following folders Configuration Templates, Configurations, Connection Templates, and Connections.

- 4 To view the contents of the Local Admin Repository folders, you can double-click them to display their contents in an editor.

Contents of the Local Admin Repository

The Local Admin Repository contains a Connection Project that references four folders, each folder contains templates or instance files that support the default repository configurations and connections that you see in the New Repository wizard and the New Workspace and New Repository Connectionwizards in Blaze Advisor.

The Admin Repository also contains the system folder that contains the templates and instances for the management features in Blaze Advisor. See ["Managing Templates and Instances in Your system Folder" on page 119](#).



Caution If you edit or delete any of the existing templates or instance files, you cannot undo your actions.

Configuration Templates

The Configuration Templates folder contains the Repository Configuration template. The configurations that are contained in the Configurations folder are instance files of the Repository Configuration template. The Repository Configuration contains the mandatory and optional classes that make it possible to define the persistence type and the optional services you want to use in a repository. If you want to make changes to the repository types you see in the New Repository wizard in Blaze Advisor, you edit the configuration instances in the Configurations folder.

Configuration Instances

The Configurations folder contains instance files of the Repository Configuration template where the persistence type and the optional services are defined. When you double-click the instance file in Repository Explorer, you can see the repository configuration parameters you see on the Details page of the creation wizard. See ["Editing a Repository Configuration in the Blaze Advisor IDE" on page 39](#).

You can edit the configuration parameters in the instance editor of the Admin Repository and any changes you make in the Admin Repository will appear immediately in the New Repository wizard in Blaze Advisor.



Note The repository configurations that you create when you use the New Repository wizard in Blaze Advisor are not persisted in the Admin Repository. However, if you discover that you have made errors in the configuration, you can edit the `com.blazesoft.rom_config.cfg` located in the repository directory.

Connection Templates

The Connection Templates folder contains a set of connection templates for the repository connections you would most frequently use in the Connect to Repository wizard in Blaze Advisor. The connection templates predefine the connection parameters you need for the type of repository connection you want to create. The connection templates support the list of connections you see in the Repository Connections window in the wizard. See ["Connecting to a Blaze Advisor Workspace or Repository" on page 52](#).

If you want to make changes to the connection types, you edit the connection instances in the Connections folder.

Connection Instances

The Connections folder contains the Default Repository Connection instance which is based on the File Repository (no versioning) Connection Template. When you create a new repository connection using the New Repository Connection, the Import, the Export, or the Manage Connections wizard in Blaze Advisor or use the command-line arguments to launch and connect to a workspace or repository, the connection is saved in the Connections folder as an instance of a connection template.

You can edit the connection parameters in the instance editor of the Admin Repository and any changes you make in the Admin Repository will appear immediately in the New Repository Connection, the Import, the Export, or the Manage Connections wizard in Blaze Advisor.

When you create a new connection and you select the option to save the workspace credentials and or the repository credentials, this information is saved in the connection instance file stored in the Admin Repository. It is also added to the repository configuration file, `com.blazesoftware.repository_config.cfg`.

If you are using a non-file BVS repository and you selected to option to allow the workspace connection credentials to be distinct from the repository connection credentials you see that two different sets of credentials are stored.

Editing Configurations and Connections Instances

You can edit the parameters of existing configuration and connection instance files and have these changes appear in the pages of the repository wizards in Blaze Advisor. For example, if you create a new connection instance it appears in the list of available connections when you invoke the Manage Connections wizard from the Project>Repository menu. You can also edit some fields within the connection templates and have these changes appear in the fields of the repository wizards. However, if you create new configuration instance files or connection templates, they will not appear in the repository wizards in Blaze Advisor.

Editing Configuration Instance Files

You can edit the configuration instance files in the Admin Repository. Any changes you make to existing configuration instance files in the Admin Repository appear in the Details page of the New Repository wizard in Blaze Advisor and can be used with any new repository you create using the repository type.

You can also edit the Details page of the wizard when you are creating a repository and you want to add or delete service options for a specific repository. However these changes are not persisted as a separate configuration instance in the Admin Repository and cannot be reused. See "["Editing a Repository Configuration in the Blaze Advisor IDE" on page 39](#).

Editing Connection Templates

You can edit the connection templates in the Admin Repository. Any changes you make in the Admin Repository, excluding the creation of new connection templates, may be displayed in the repository wizards in Blaze Advisor. The following topics provides examples of how you can edit a connection template.

Changing the Password Display in an Existing Connection Template

You can change how user input is displayed in the password field in the repository wizards.

To edit the display of the password field

- 1 If you have not already done so, connect to the Admin Repository.
See "[Connecting to the Local Admin Repository](#)" on page 132.
- 2 Double-click to open the **Connection Templates** folder and choose the repository connection template you want to view in a editor.
- 3 In **Display Format** field, locate the `com.blazesoftware.repository_password` display placeholder and double-click it to open the Placeholder Wizard.
- 4 The **Placeholder Wizard** appears with the `com.blazesoftware.repository_password:string` provider value preselected.
- 5 In the **Hints for Application Generation** window, edit the **Key Echo** field to use Password Style or Normal Echo and click OK.
If you want the user input to display as a text string, use the **Normal Echo** setting. However, if you want the user input to display to be hidden by a set of asterisks, `(*****)`, choose **Password Style**.
- 6 Click one of the following items:
 - **Finish** to register your changes and return to the template editor.
 - **Cancel** to exit the template editor without saving your changes.

Editing the Directory Path in an Existing Connection Template

You change the default directory path for a repository or workspace on the connection parameters page of a creation or connection wizard.

To edit a default directory location

- 1 If you have not already done so, connect to the Admin Repository.
See "[Connecting to the Local Admin Repository](#)" on page 132.
- 2 Double-click to open the **Connection Templates** folder and choose the repository connection template you want to view in a editor.

- 3 In the **Value Holders and Arguments** window, expand the `com.blazesoftware.repository.workspacefolder` or `com.blazesoftware.repository.repositoryfolder` value holder and click **Jump to the template/provider** editor icon  located to the right of the Directory Path Provider.
- 4 Edit the directory path in the **Default** field and click Enter and Save.
The edited text appears in the Repository Location or Local Folder text field.
Any value holder of type Directory Path Provider appears in the Blaze Advisor IDE and in a generated rule maintenance application as a text field and a Browse button.
- 5 Save your changes.

Editing Connection Instance Files

Connection instance files are located in the Connections folder and can be edited in the instance editor in the Admin Repository. You edit a connection instance if you want to change the values of the connection parameters for a given repository connection. Any changes you make cannot be reverted and are available to the repository wizards.

You can edit the connection parameters of every connection instance except for the Default Repository Connection that is provided with your Blaze Advisor installation.

Creating a New Connection Instance to use a Database Private Workspace

By default, when you create a BVS repository with a private workspace, the workspace is of file type. If you want to create a File or Database repository with a private Database workspace, you need to create a connection instance for the File or Database repository and specify a File Repository (Generic SCM versioning) connection for it in the Admin Repository. You can choose a File or Database type as your File Repository (Generic SCM versioning) connection type. When you return to the Blaze Advisor IDE and use the creation wizard to create a BVS Database repository that uses a private workspace, you leave the Local Folder field blank. When you use the connection wizard to create the workspace connection, you select the new connection instance you created in the Admin Repository and use the Advanced button in the wizard to open a page in the wizard that allows you to enter the Repository and Workspace connections.



Note If you create a BVS repository with a private Database workspace, and you need to update the system templates, you may need to convert your workspace to a File repository. See “[Verifying Blaze Advisor Versioning Commands](#)” on page 230.

Manually Creating a Connection Based on an Existing Connection Instance

To create a new connection, you can copy an existing instance file, paste it into Connections folder, and edit the connection parameters. After you save the new connection, it appears in the list of available connections when you invoke the ManageConnections wizard from the Repository menu.

Deleting Connection Instance Files

Any connections that are no longer being used can be deleted using the Manage Connections wizard in Blaze Advisor IDE or from the Admin Repository.

You can delete connections from the Manage Connections wizard that you invoke from the Repository menu, see "[Deleting a Blaze Advisor Repository or Workspace Connection](#)" on page 60.

You can also delete connections from the Admin Repository. If you delete a connection instance, you receive a message informing you that the deletion cannot be undone. After you delete the connection instance, you will no longer see it in the list of available connections when you invoke the New Repository Connection, the Import, the Export, or the Manage Connections wizard from the Blaze Advisor IDE menu.

Synchronizing the system Folder

You use the `NdRomAdminUtil` utility to synchronize the contents of the system folder in your repository with the system folder in the Admin Repository of your Blaze Advisor installation. You may want to synchronize your system folder if you have altered the templates and instances in the system folder of your repository and wish to use the default contents provided by Blaze Advisor.

If you have modified the templates and instances in your system folder, see "[How Modifications to Templates and Instances in the system Folder are Treated by the Synchronization Process](#)" on page 137.

The steps you must complete to synchronize your system folder to use the default system folder are:

- ["Prerequisites for Synchronizing the system Folder of a Non-versioned Repository" on page 138](#)
- ["Prerequisites for Synchronizing the system Folder of a Versioned Repository" on page 138](#)
- ["Running the Utility to Synchronize a system Folder" on page 139](#)
- ["Updating Workspaces Connected to a Synchronized Repository" on page 142](#)

If you are using a versioned repository that has more than one private or local workspace that connects to it, you need to run the `NdRomAdminUtil` utility to update those workspaces after you have synchronized your system folder. The `update` utility command updates the system folder in those workspaces.

How Modifications to Templates and Instances in the system Folder are Treated by the Synchronization Process

If you have made any changes to the templates and instances in the system folder of your repository, need to be aware of how synchronization of your system folder affects those changes. For information on how to synchronize your system folder, see "[Synchronizing the system Folder](#)" on page 137.

- Additions to your system folder are preserved.
For example, if you added a Query template or instance, it is preserved.
- Modifications to the management property set are preserved.
For example, if you edit a Management Property Type, the edit is preserved.
- Deletions of entities in the system folder are not preserved.
For example, if you deleted the Management Property Folder, it will be replaced by a new version.
- Modification to non-management property entities are not preserved.
For example, if you edited the Query Template, your edits are overwritten.

When you synchronize the system folder in your repository, you will be able to use the default management features available the templates and instances provided with your Blaze Advisor installation.

Prerequisites for Synchronizing the system Folder of a Non-versioned Repository

You can use the `NdRomAdminUtil` utility and the `synchronizeSystemFolder` command to synchronize the system folder of a non-versioned repository with the system folder in the Admin Repository located in the `<ADVISOR_HOME>/lib` directory of your Blaze Advisor installation.

Before you can run the utility, prior to using the `NdRomAdminUtil` utility, you need to perform the following tasks:

- Write an XML file (`.cfg`) for the connection to the Admin Repository in your Blaze Advisor installation. See "[Repository Connection Configuration File Examples](#)" on page 171.
- Write an XML file (`.cfg`) for the repository connection configuration. You need to write a repository connection configuration file for the type of repository you want to synchronize, see "[Repository Connection Configuration File Examples](#)" on page 171.

After you have completed these tasks, you can run the `NdRomAdminUtil` utility, see "[Running the Utility to Synchronize a system Folder](#)" on page 139.

Prerequisites for Synchronizing the system Folder of a Versioned Repository

You can use the `NdRomAdminUtil` utility and the `synchronizeSystemFolder` command to synchronize the system folder of a versioned repository with the system folder in the Admin Repository located in the `<ADVISOR_HOME>/lib` directory of your Blaze Advisor installation.

If you are using a versioned repository, you synchronize the system folder in a workspace and then commit the change to the repository. If you have more than one workspace

that needs to connect to the repository, you need to update these workspaces after completing the synchronization of the system folder and committing the configuration changes. To prepare a versioned repository for synchronization, you need to ensure that you do not have any items checked out and that all the files are updated in the repository. Please check in or cancel the check out of any items prior to synchronizing the system folder. If you are using a BVS repository with a private workspace or a File CVS repository, you also must ensure that your workspace is updated by using the Update command on the Repository Explorer toolbar in the Blaze Advisor IDE. If you are using a File BVS shared repository or a non-versioned repository, you need to use the Refresh All command in the Blaze Advisor IDE to ensure that your workspace is updated.

Before you can run the utility, you need to perform the following tasks:

- Write an XML file (.cfg) for the connection to the Admin Repository in your Blaze Advisor installation. See "[Repository Connection Configuration File Examples](#)" on page 171.
- Write an XML file (.cfg) for the workspace connection configuration. You need to write a workspace connection configuration file for the type of repository you want to migrate, see "[Repository Connection Configuration File Examples](#)" on page 171.
Please note that in the case of a versioned repository, you write a connection configuration file for the workspace instead of the repository.
- If you have several local or private workspaces that need to connect to the repository, you need to run the utility with the update command. See "[Updating Workspaces Connected to a Synchronized Repository](#)" on page 142.

After you have completed these tasks, you can run the NdRomAdminUtil utility, see "[Running the Utility to Synchronize a system Folder](#)" on page 139.

Running the Utility to Synchronize a system Folder

After you have written the connection configuration files for the Admin Repository and the workspace or repository, you are ready to run the NdRomAdminUtil utility. You use the synchronizeSystemFolder command when you run the NdRomAdminUtil utility to synchronize the system folder in your repository with the system folder in the Admin Repository in your Blaze Advisor installation.

If you are using a versioned repository, you need to make sure that you do not have any items checked out in your repository and that you have updated your workspace and your system folder. Please use the Blaze Advisor version you used to create the workspace to check in any items, cancel the check out of any items, and/or update the workspace.

- If you are using a File or Database BVS repository with a private workspace or a File CVS, Subversion, and Clearcase repository with a local workspace, you also must ensure that your workspace is updated by using the Update command.
- If you are using a File BVS repository with a shared workspace or a non-versioned repository, you need to update your workspace by using the Refresh All command.

- If you have any files checked out of your system folder, you need to check them back in, prior to running the utility.

! **Important** If you fail to up-to-date your repository correctly, the synchronization of your system folder may not be successful.

If you are using a versioned repository, we recommend that you use the `-autocheckin` option when you run the utility to automatically check in any updates to the system folder. If you do not specify this option, you will need to check in the system folder changes for each file in the system folder separately.

If you are using a versioned repository that has more than one private or local workspace that connects to it, you need to run the `NdRomAdminUtil` utility to update the system folders in those workspaces after you have synchronized your system folder. See [“Updating Workspaces Connected to a Synchronized Repository” on page 142](#).

To synchronize the system folder using the `NdRomAdminUtil` utility:

Command, Argument, or Option	Description
<code>synchronizeSystemFolder</code>	Command to synchronize the system folder in your existing repository with the system folder in the Admin Repository of the latest Blaze Advisor installation on your machine.
<code>-workspaceConnection</code>	Argument takes the pathname to the location of the workspace connection configuration (.cfg). You need to write a connection configuration file for the type of repository you are using. If you are using a versioned repository with a local or private workspace, you need to write a connection configuration file for the workspace type. See: <ul style="list-style-type: none">■ “Repository Connection Configuration File Examples” on page 171■ “Writing a Connection Configuration File for a Private or a Local Workspace” on page 185 This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
<code>-systemconnection</code>	Argument takes the pathname to the location of the system folder connection configuration (.cfg). See “Repository Connection Configuration File Examples” on page 171 . This file contains information such as the name of the repository which in this case is Admin Repository, the location of the Admin Repository and the user name.
<code>-l</code>	Option used to specify whether you want the action to be performed only on the local directory. If you don't use this option, the default behavior is for the action to be performed recursively.
<code>-autoCheckIn</code>	Option used to with a versioned repository to automatically check in the system folder contents as soon as the synchronization process is completed.

Command, Argument, or Option	Description
-verbose	Option prints out all messages into the command console.
-directoryLocation	<p>Option takes a directory location relative to the root directory in the workspace where the system folder you want to synchronize is located.</p> <p>Example: "/system"</p> <p>Important When using the synchronizeSystemFolder command, the value for the -directoryLocation argument needs to reference the system folder in the workspace.</p>

To run the utility to synchronize the system folder

- 1 Open a command prompt at <ADVISOR_HOME>
- 2 Set the Blaze Advisor environment using setenv.
- 3 Run the NdRomAdminUtil utility from a command prompt and enter the command and arguments directory in the console or by using a batch file.

This is an example of running the NdRomAdminUtil utility for a versioned repository to synchronize system folders using a command prompt:

```
C:\Blaze\Advisor\bin> setenv
C:\Blaze\Advisor\bin>java
com.blazesoftware.template.repository.admin.NdRomAdminUtil
synchronizeSystemFolder -systemConnection "c:/Repo_Connections/
ConnectionFiles/sysAdmin.cfg"
-workspaceConnection "c:/Repo_Connections/ConnectionFiles/
DefaultVerWksp.cfg"
-directoryLocation "/system" -autocheckin -verbose
```

This is an example of a batch file (.bat) using the NdRomAdminUtil utility for a versioned repository to synchronize system folders:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil
synchronizeSystemFolder -systemConnection "c:/Repo_Connections/
ConnectionFiles/sysAdmin.cfg"
-workspaceConnection "c:/Repo_Connections/ConnectionFiles/
DefaultVerWksp.cfg"
-directoryLocation "/system" -autocheckin -verbose
```

When you have successfully synchronized the system folder and are using the -autocheckin option for a versioned repository, you see the following message in your command prompt:

```
...
Entry: /system/Management Properties Library Folder/Management Property
Category
    Templates/Srl Entities/SRL Ruleset has been Merged
Entry: /system/Management Properties Library Folder/Management Property
Category
    Templates/Srl Entities/SRL Function has been Merged
Entry: /system/Management Properties Library Folder/Management Property
Category
```

```
Templates/Entry has been Merged
Entry: /system/Management Properties Library Folder/Management Property
Types/
source reference has been Added
Entry: /system/Management Properties Library Folder/Management Property
Types/
Test Roles has been Added
Entry: /system/Management Properties Library Folder/Management Property
Templates/Test Role has been Added
Entry: /system/Management Properties Library Folder/Management Property
Templates/Source Reference has been Added
Entry: /system/Management Properties Library Folder/Management Property
Templates/Test Case has been Added
Entry: /system/Queries Library Folder/Builder Queries/Standard Query
Folder/
Common/BuiltIns/Entity Type Provider has been Updated
Entry: /system/Queries Library Folder/Builder Queries/Standard Query
Folder/
Templates/Text Content/Text Content Filter has been Updated
Entry: /system/Queries Library Folder/Builder Queries/Standard Query
Folder/
Templates/Historical Information/Historical Information Filter has been
Updated
Entry: /system/Queries Library Folder/Builder Queries/Standard Query
Folder/
Templates/Management Properties/Management Property Filter has been
Updated
Entry: /system/Queries Library Folder/Builder Queries/Standard Query
Folder/
Templates/Cross Reference/Cross Reference Filter has been Updated
Entry: /system/Queries Library Folder/Builder Queries/Standard Query
Folder/
Standard Query Template has been Updated
Entry: /system/Queries Library Folder/Business Queries/Standard Business
Query
has been Updated
Entry: /system/Query Object Model Library Folder/Query Bom has been
Updated
Command <synchronizeSystemFolder> is executed successfully
```

- 4 If you are using a non-versioned repository, you are ready to create a new connection to the synchronized/migrated repository in Blaze Advisor.
If you are using a versioned repository that has more than one private or local workspace, you need to run the `NdRomAdminUtil` utility to update the system folders in those workspaces. See "[Updating Workspaces Connected to a Synchronized Repository](#)" on page 142.

Updating Workspaces Connected to a Synchronized Repository

If you have synchronized the system folder of a versioned repository with a private or local workspace, checked in the system folder changes, and you have committed the changes to the repository configuration, you now need to update any workspaces that connect to the repository, including the workspace you used for the synchronization process. You update each workspace so that any deleted items are removed from your

workspace and the contents in your system folder are the same as the contents in the system folder in the repository.

There are several methods you can use to perform an update:

- Use the update utility command as explained below.
- Create and populate a new workspace in Blaze Advisor.
- Connect to the existing workspace in the Blaze Advisor IDE and update the workspace using the Update command on the Repository Explorer toolbar.



Tip If you create a new workspace connection in Blaze Advisor and the workspace is not automatically populated when you connect to it, select the root directory located in the Repository Explorer and click the Update icon on the Repository toolbar to populate your workspace.

To update the workspace, you can also use the `NdRomAdminUtil` utility:

Command, Argument, or Option	Description
update	Command to update changes from the repository to any workspaces connect to it.
-workspaceConnection	<p>Argument takes the pathname of the workspace connection configuration file. If you have more than one existing workspace that needs to connect to the migrated repository, you need to write a connection configuration file for each workspace. See</p> <ul style="list-style-type: none"> ■ “Repository Connection Configuration File Examples” on page 171 ■ “Writing a Connection Configuration File for a Private or a Local Workspace” on page 185 <p>Alternatively, you can use the name of the repository connection you created in Blaze Advisor. See “Entering a Symbolic Name or a Pathname as an Argument Value” on page 166.</p>
-entryLocation	Argument takes as its value the location where you want to start your update. For example, if you want the update to occur from the root directory, you would use “/”.
-verbose	Option prints out all messages into the command console.
-l	Option allows the update to be performed only on a local directory. The default behavior is for the update action to occur recursively.

To run the utility to update a workspace

- 1 If you have not already done so, open a command prompt at <ADVISOR_HOME>
- 2 Set the Blaze Advisor environment using `setenv`.

- 3 Run the `NdRomAdminUtil` utility from a command prompt and enter the command and arguments directory in the console or by using a batch file.

This is an example of running the `NdRomAdminUtil` utility for a versioned repository to update a workspace using a pathname as the value for the

-workspaceConnection argument:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil update  
-workspaceConnection "c:/Blaze/Advisor/RepoWksp/Workspace.cfg" -  
entryLocation "/"  
-verbose
```

This is an example of using a symbolic name as a value for the -workspaceConnection argument:

```
java -DADVISOR_HOME="c:/Blaze/Advisor"  
com.blazesoftware.template.repository.admin.NdRomAdminUtil update  
-workspaceConnection "NewRepoConn" -entryLocation "/"  
-verbose
```

- 4 When you have successfully updated the changes to the workspace, you see the following message in your command prompt:

```
...  
Command <update> is executed successfully
```

If you are using a private or local workspace, you can now use the New Repository Connection, the Import, the Export, or the Manage Connections wizard to create a new connection in Blaze Advisor.

Changing an Existing Repository Configuration

If you want to change the repository configuration by adding or removing services from your repository, you need edit the `romConfig.cfg` file from the `<ADVISOR_HOME>/lib` directory of your Blaze Advisor installation and use its location as an argument value. The information you need to edit in this file depends on whether you are using a non-versioned repository or a versioned file repository with a shared workspace or a versioned file and database repository with a private or local workspace.



Important We recommend that you take a copy of the `romConfig.cfg` to edit and use to change the repository configuration.

In the batch file or command prompt, you reference the path to these files as argument values. See ["Running the NdRomAdminUtil Utility" on page 166](#).

Command, Argument, or Option	Description
changeConfig	Command change the workspace configuration.
-workspaceConnection	<p>Argument takes the path to the location of the workspace connection configuration file (.cfg). See "Repository Connection Configuration File Examples" on page 171.</p> <p>This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.</p> <p>Alternatively, you can use a symbolic name for the workspace connection. See "Entering a Symbolic Name or a Pathname as an Argument Value" on page 166.</p>
-configuration	<p>Argument takes the path to the location of the repository configuration file (.cfg). This configuration file needs to include the workspace connection. See "Writing a Repository Configuration File" on page 191.</p>
-verbose	Option prints out all details messages into the command console.

For information on using the `changeConfig` command with the utility to change the configuration of a non-versioned or versioned repository, see:

- ["Changing the Configuration of a Non-versioned Repository" on page 146](#)
- ["Changing the Configuration of a Versioned Repository" on page 148](#)

About the ROM Configuration File Provided with Your Blaze Advisor Installation

The `romConfig.cfg` file that is located in the `<ADVISOR_HOME>/lib` directory of your Blaze Advisor installation is the Repository Object Model configuration used when you want to change the configuration of an existing workspace or repository using the `NdRomAdminUtil` utility. You change the configuration of a repository or workspace when you want to add or remove services. If you are using a versioned repository, you change the configuration in a workspace and then commit the changes to the repository.

When a repository or workspace is created, two configuration files and their related `.attbs` files are added to the root directory. These configuration files, `com_blazesoft_rom_config.cfg` and `com_blazesoft_repository_config.cfg` contain references to classes that provide the services and storage mechanism for a repository. You cannot use these files to change the configuration of an existing repository. The `romConfig.cfg` contains a combination of the tags and elements found in the `com_blazesoft_rom_config.cfg` and `com_blazesoft_repository_config.cfg`. This file can be edited and then referenced in a command prompt or batch file when running the `NdRomAdminUtil` utility.

Changing the Configuration of a Non-versioned Repository

You can change the configuration of an existing non-versioned repository by manually editing the `romConfig.cfg` file that is located in the `<ADVISOR_HOME>/lib` directory of your Blaze Advisor installation. You edit this configuration file by adding or removing the corresponding tags for services and use its location as an argument value when running the `NdRomAdminUtil` utility with the `changeConfig` command. The `NdRomAdminUtil` utility is a command-utility used for many administrative tasks related to repositories and workspaces. In a non-versioned repository because the workspace and the repository share the same physical location, after you have completed these two steps, you are ready to connect to the repository.

For more information about running the `NdRomAdminUtil` utility, see “[Importing or Exporting Projects and Workspace Contents Using the Utility](#)” on page 65.



Important We recommend that you take a copy of the `romConfig.cfg` to edit and use to change the repository configuration.

To change the configuration of a non-versioned repository

- 1 Edit the `romConfig.cfg` configuration file located in the `\lib` directory of your Blaze Advisor installation.

In this example, we are adding an authorization manager:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RomConfig>
    <AllowUnusedValues> true </AllowUnusedValues>
    <PromProjectReleaseManagerConfig>
        <PromProjectReleaseManagerFactory>
            <JavaName>
                com.blazesoftware.template.repository.impl.NdDefaultPromProjectReleaseManager
            </JavaName>
        </PromProjectReleaseManagerFactory>
    <RomExtractorConfig>
        <RomExtractorFactory>
            <JavaName>
                com.blazesoftware.template.repository.deploy.NdPromDefaultEntityContentExtractor
            </JavaName>
        </RomExtractorFactory>
    </RomExtractorConfig>
    </PromProjectReleaseManagerConfig>
    <RomConnectionManagerConfig>
        <ConnectionMode> 0 </ConnectionMode>
        <RomConnectionManagerFactory>
            <JavaName>
                com.blazesoftware.template.repository.impl.NdDefaultRomConnectionManager
            </JavaName>
        </RomConnectionManagerFactory>
    </RomConnectionManagerConfig>
    <RomQueryManagerConfig>
```

```

<RomQueryManagerFactory>
  <JavaName>
    com.blazesoft.template.repository.query.NdRomDefaultQueryMan
ager
  </JavaName>
</RomQueryManagerFactory>
</RomQueryManagerConfig>
<RomSchemaManagerConfig>
  <RomSchemaManagerFactory>
    <JavaName>
      com.blazesoft.template.repository.impl.NdDefaultRomSchemaMan
ager
    </JavaName>
  </RomSchemaManagerFactory>
</RomSchemaManagerConfig>
<RepositoryConfig>
  <RepositoryAuthorizationManagerFactory>
    <JavaName>
      examples.CustomAuthorizationManager
    </JavaName>
  </RepositoryAuthorizationManagerFactory>
</RepositoryConfig>
</RomConfig>

```

For more information about repository configuration files, see [“Writing a Repository Configuration File” on page 191](#).

- 2 To run the `NdRomAdminUtil` utility, you need to reference a connection configuration file for the repository or workspace. You need to write a connection configuration file (`.cfg`) for the repository if you originally created the repository using the Blaze Advisor IDE. If you used the utility to create the repository, you can reuse the connection configuration file.

Example of a File repository connection configuration file:

```

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

<RepositoryConnection>
  <Factory>
    com.blazesoft.repository.file.NdFileRepositoryConnection
  </Factory>
  <RepositoryFolder> C:\Repositories\fileRepo1</RepositoryFolder>
  <User>user1</User>
  <RepositoryName>fileRepo1 </RepositoryName>
</RepositoryConnection>

```

For more information about repository and workspace connection configuration files, see [“Repository Connection Configuration File Examples” on page 171](#).

- 3 Run the `NdRomAdminUtil` utility with the `changeConfig` command.

In this example we use a batch file containing the following commands:

```

java com.blazesoft.template.repository.admin.NdRomAdminUtil changeConfig
  -workspaceConnection "c:/Repo_Connections/Connections/"

```

```
DefaultFileRepo.cfg" -configuration "c:/Blaze/Advisor/lib/romConfig.cfg"
-verbose
```

In this example, when the configuration is changed by running the `NdRomAdminUtil` utility with the `changeConfig` command, there are additional tags for the authorization manager added to the `com.blazesoftware.repository_config.cfg` file located in the root directory.

Changing the Configuration of a Versioned Repository

If you want to change the configuration of a versioned repository, in addition to manually editing the `romConfig.cfg` located in the `<ADVISOR_HOME>/lib` directory of your Blaze Advisor installation and running the `NdRomAdminUtil` utility with the `changeConfig` command, you need to commit the changes to the repository configuration.

When you want to change the configuration of an existing versioned repository, you make the changes to the workspace and then commit the to the repository by running the `NdRomAdminUtil` utility a second time with the `commitConfig` command. If you have any other workspaces that connect to the repository, you need to run the utility with the `updateConfig` command so that the configuration files in those workspaces are updated.

Changes to a versioned repository need to be made this way because the workspace and the repository are considered to be separate objects. In a file repository with a shared, or a file and database private or local workspace, you make the changes in a workspace and then commit changes to the repository configuration.



Important We recommend that you take a copy of the `romConfig.cfg` to edit and use to change the repository configuration.

To change the configuration of an existing workspace

- 1 Edit the `romConfig.cfg` configuration file located in the `<ADVISOR_HOME>/lib` directory of your Blaze Advisor installation.

In this example the `romConfig.cfg` file contains the connection configuration for the workspace that is connects to the repository where you want to add an authorization manager.

```
<?xml version="1.0" encoding="UTF-8" ?>

<RomConfig>
    <AllowUnusedValues> true </AllowUnusedValues>
    <PromProjectReleaseManagerConfig>
        <PromProjectReleaseManagerFactory>
            <JavaName>
                com.blazesoftware.template.repository.impl.NdDefaultPromProjectRel
            ease
            Manager
            </JavaName>
        </PromProjectReleaseManagerFactory>
        <RomExtractorConfig>
            <RomExtractorFactory>
                <JavaName>
                    com.blazesoftware.template.repository.deploy.NdPromDefault
                </JavaName>
            </RomExtractorFactory>
        </RomExtractorConfig>
    </PromProjectReleaseManagerConfig>
</RomConfig>
```

```

        EntityContentExtractor
    </JavaName>
</RomExtractorFactory>
</RomExtractorConfig>
</PromProjectReleaseManagerConfig>
<RomConnectionManagerConfig>
    <ConnectionMode> 0 </ConnectionMode>
    <RomConnectionManagerFactory>
        <JavaName>
            com.blazesoftware.template.repository.impl.NdDefaultRomConnection
Manager
        </JavaName>
        </RomConnectionManagerFactory>
        </RomConnectionManagerConfig>
<RomQueryManagerConfig>
    <RomQueryManagerFactory>
        <JavaName>
            com.blazesoftware.template.repository.query.NdRomDefaultQueryMan
ager
        </JavaName>
        </RomQueryManagerFactory>
    </RomQueryManagerConfig>
    <RomSchemaManagerConfig>
        <RomSchemaManagerFactory>
            <JavaName>
                com.blazesoftware.template.repository.impl.NdDefaultRomSchemaMan
ager
            </JavaName>
            </RomSchemaManagerFactory>
        </RomSchemaManagerConfig>

<RepositoryConfig>
<! AtticRepository tags are auto-generated for most repository types
and its value of true is only relevant for a BVS shared repository -->
<AtticRepository> false </AtticRepository>
<RepositoryAuthorizationManagerFactory>
    <JavaName> examplesForAuthorization.CustomAuthorizationManager
    </JavaName>
</RepositoryAuthorizationManagerFactory>
<RepositoryVersionManagerConfig>
    <PrivateWorkspaceSupported> true </PrivateWorkspaceSupported>
<RepositoryVersionManagerFactory>
    <JavaName>
        com.blazesoftware.repository.generic.version.NdNativeRepository
        VersionManager
    </JavaName>
</RepositoryVersionManagerFactory>
<VersioningRepositoryConnection>
    <Factory>
        com.blazesoftware.repository.generic.version.
        NdNativeVersioningRepositoryConnection
    </Factory>
    <Impersonate> true </Impersonate>
    <PersistCredentials> false </PersistCredentials>
    <RepositoryConnection>
        <Factory>

```

```
com.blazesoftware.repository.file.NdFileRepositoryConnection
</Factory>
<CredentialsFactory>
<JavaName> com.blazesoftware.repository.base.
    NdUserPasswordCredentials</JavaName>
</CredentialsFactory>
<Name> Repository Connection Name </Name>
<RepositoryFolder> C:\60kits\3692\CM </RepositoryFolder>
<RepositoryName> CM </RepositoryName>
<User> user1 </User>
</RepositoryConnection>
<RepositoryName> CM </RepositoryName>
<User> user1 </User></VersioningRepositoryConnection>
<WorkspaceUser> user1 </WorkspaceUser>
</RepositoryConfig>
</RepositoryVersionManagerConfig>
</RomConfig>
```

- 2 Run the `NdRomAdminUtil` utility using a batch file or command prompt.

Example of running the utility using a command prompt:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil changeConfig
-workspaceConnection "c:/Repo_Connections/ConnectionFiles/FileWksp.cfg"
-configuration "c:/Blaze/Advisor/lib/romConfig.cfg" -verbose
```

Next, you need to commit the changes to your repository. See "["Committing a Configuration Change from a Workspace to a Repository" on page 150](#)".

Committing a Configuration Change from a Workspace to a Repository

If the configuration of your repository has been changed to include or remove services, and you are using a versioned repository, you need to commit these changes to your repository.

If you have any other existing private or local workspaces that connect to the repository, you need to update the configuration of these workspaces. See "["Updating a Configuration Change from a Repository to a Workspace" on page 151](#)".

To commit the changes, you use the `NdRomAdminUtil` utility:

Command, Argument, or Option	Description
commitConfig	Command to commit changes from the workspace configuration to the repository.

Command, Argument, or Option	Description
-workspaceConnection	<p>Argument takes the pathname of the workspace connection configuration file. Here you can use the workspace connection configuration file you used earlier to synchronize the system folder. See “Writing Connection Configuration Files” on page 169.</p> <p>Alternatively, you can use the name of the repository connection you created in Blaze Advisor. See “Entering a Symbolic Name or a Pathname as an Argument Value” on page 166.</p>
-verbose	Option prints out all messages into the command console.

To run the utility to commit the changes to the repository configuration

- 1 If you have not already done so, open a command prompt at <ADVISOR_HOME>
- 2 Set the Blaze Advisor environment using `setenv`.
- 3 Run the `NdRomAdminUtil` utility from a command prompt and enter the command and arguments directory in the console or by using a batch file.

This is an example of running the `NdRomAdminUtil` utility to commit the repository configuration changes for a versioned repository. The `-workspaceConnection` argument takes a pathname to the workspace connection configuration file.

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil commitConfig
    -workspaceConnection "c:/Blaze/Advisor/RepoWksp/Workspace2.cfg"
    -verbose
```

This is an example of using a symbolic name for the value of the `-workspaceConnection` argument:

```
java -DADVISOR_HOME="c:/Blaze/Advisor"
    com.blazesoft.template.repository.admin.NdRomAdminUtil commitConfig
    -workspaceConnection "NewRepoConn"
    -verbose
```

When you have successfully commit the changes to the repository configuration, you see the following message in your command prompt:

```
...
Command <commitConfig> is executed successfully
```

If you are using a private or local workspace, you can now update the configuration of your workspaces. See “[Updating a Configuration Change from a Repository to a Workspace](#)” on page 151.

Updating a Configuration Change from a Repository to a Workspace

If you change the configuration for your versioned repository and you have other workspaces that need to connect to the repository, you can update the configurations of these corresponding workspaces using the `NdRomAdminUtil` utility. If you used the utility to create the workspaces, you can reuse the workspace connection configuration files you wrote earlier. If you created the workspaces in the Blaze Advisor IDE, you need to

write connection configuration files for these workspaces. The workspace connection configuration files contains the repository connection information. See ["Writing a Connection Configuration File for a Private or a Local Workspace" on page 185](#).

To update the configuration, you use the `NdRomAdminUtil` utility:

Command, Argument, or Option	Description
<code>updateConfig</code>	Command to update the workspace configuration with the latest version of the configuration from the repository.
<code>-workspaceConnection</code>	Argument takes the pathname of the workspace connection configuration file. If you have more than one existing workspace that needs to connect to the migrated repository, you need to write a connection configuration file for each workspace. See "Writing Connection Configuration Files" on page 169 . Alternatively, you can use the name of the repository connection you created in Blaze Advisor. See "Entering a Symbolic Name or a Pathname as an Argument Value" on page 166 .
<code>-verbose</code>	Option prints out all details messages into the command console.

This is an example of running the `NdRomAdminUtil` utility for a versioned repository to update the configuration of a workspace using a pathname as the value for the `-workspaceConnection` argument:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil updateConfig  
-workspaceConnection "c:/Blaze/Advisor/RepoWksp/Workspace.cfg" -verbose
```

This is an example of using a symbolic name as a value for the `-workspaceConnection` argument:

```
java -DADVISOR_HOME="c:/Blaze/Advisor"  
com.blazesoftware.template.repository.admin.NdRomAdminUtil updateConfig  
-workspaceConnection "NewRepoConn" -verbose
```

Loading Custom Classes

If you have a configuration that has a dependency on an external class, such as a custom class that adds a service to a repository or a feature to a rule service, you can edit the corresponding factory tags to add the location of the assembly or class. You can specify the location of assemblies and/or classes so that the services or features can be used by the .NET and/or Java platforms of Blaze Advisor.

- The following tags enable the dynamic object loader to locate the assembly and/or class: repository configuration (`com.blazesoftware.repository_config.cfg`)
- rule server configuration (`.server`)

- brUnit test case configuration (`testSuiteConfig` or `promTestSuitConfig`)

The brUnit unit tests cases that have a dependency on external classes use tags with slightly different names. These tags are used to by the dynamic loader in the same way to locate classes. See "TestClassLoaderFactory" in *Reference.pdf*.

In some cases, the tags are automatically added to a file, for example when you create a repository in the Blaze Advisor IDE and specify a custom class and/or assembly, you see the tags in the repository configuration. See "[Editing a Repository Configuration in the Blaze Advisor IDE](#)" on page 39.

All factory tags support the following child tags for providing the location of assemblies and/or custom classes in addition to the `<JavaName>` tag that specifies the class name to be instantiated by the factory:

Tag Name	Description
<AssemblyConfig/>	<p>Specifies the information needed to load the assembly that contains the class to be instantiated by the factory.</p> <p>It is possible to write a configuration file that supports both .NET and Java. You can have the <code><AssemblyConfig/></code> used for the .NET version and the <code><ClasspathConfig/></code> used for the Java version present, in the same configuration file.</p> <p>There can be zero or more <code><ReferencedAssemblyConfig></code> tag within a set of <code><AssemblyConfig/></code> tags.</p> <p>Nested tags:</p> <ul style="list-style-type: none"> ■ (Required) <code><FullName></code> The full name of the assembly to be loaded before loading the class specified in the <code><JavaName></code> tag of the factory tag. ■ (Optional) <code><InGAC></code> A boolean value of true that specifies that the assembly is registered in the Global Assembly Cache (GAC). If the tag is not specified, the default is false. ■ (Optional) <code><HintPath></code> The hint path for the location of the assembly. Use this tag to load the assembly when the a configuration is loaded using the Blaze Advisor IDE, or an Advisor utility such as the NdRomAdminUtil or the NdAdvisorProjectCompiler. The specified path can be either absolute or relative (recommended). <p>When a configuration is obtained from a file, the path is interpreted as relative to the current working directory of the application.</p>
<ReferencedAssemblyConfig/>	<p>(Optional) Specifies the information needed to load any assemblies referenced directly or indirectly by the assembly specified in the <code><AssemblyConfig/></code> tag.</p> <p>The nested tags for a set of <code><ReferencedAssemblyConfig/></code> tags are the same the nested tags in set of a <code><AssemblyConfig/></code> tags.</p> <p>There can be zero or more <code><ReferencedAssemblyConfig/></code> tags within a set of <code><AssemblyConfig/></code> tags but there cannot be any <code><ReferencedAssemblyConfig/></code> tags within a set of <code><ReferencedAssemblyConfig/></code> tags.</p>

Tag Name	Description
<ClasspathConfig/>	<p>(Optional) Specifies the information needed to load the class to be instantiated by the factory tag.</p> <p>It is possible to write a configuration file that supports both .NET and Java. You can have the <AssemblyConfig/> used for the .NET version and the <ClasspathConfig/> used for the Java version present, in the same configuration file.</p> <p>Nested tag:</p> <p><ClasspathEntry/> Specifies the classpath entry needed to load the class specified in the <JavaName/> tag of the factory tag. The classpath can be a directory or the location of a jar file. The specified path can be either absolute or relative (recommended).</p> <p>When a configuration is obtained from a file, the path is interpreted as relative to the current working directory of the application.</p> <p>There can be one or more <ClasspathEntry/> tags in a <ClasspathConfig/>.</p>
<KeepAbsolutePaths/>	<p>(Optional) Child tag of the factory tag. A boolean value of true specifies that any absolute paths defined within a <HintPath/> or <ClasspathEntry/> tag will remain absolute when the configuration is written out to an XML configuration file.</p> <p>If this tag is not present, it has a value of false (default) any absolute paths in the <HintPath/> or <ClasspathEntry/> tags are converted to relative paths if possible when the configuration is written out to an XML configuration file.</p>

Example

```

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

<RepositoryConfig>
    <! AtticRepository tags are auto-generated for most repository types and
    its value of true is only relevant for a BVS shared repository -->
        <AtticRepository> false </AtticRepository>

        <RepositoryAuthorizationManagerFactory>

            <AssemblyConfig>
                <FullName> authorizationManager, Version=1.0.0.0, Culture=neutral,
                    PublicKeyToken=eefacf401e1a9bef </FullName>
                <HintPath>
                    ..\..\authorizationManager\dotNET\authorizationManager.dll
                </HintPath>
                <InGAC> false </InGAC>
            </AssemblyConfig>

            <ClasspathConfig>
                <ClasspathEntry> ..\..\classes </ClasspathEntry>
            </ClasspathConfig>

            <JavaName>
examplesAuthorizationManager.ExampleRepositoryAuthorizationManager
            </JavaName>
        </RepositoryAuthorizationManagerFactory>
    ...

```

Creating Self-contained Configurations

You can add the following advanced tags as child tags to create configurations that are fully self-contained and that can be loaded using the generic XML configuration loader, `com.blazesoft.dynobj.xml.NdXmlConfigurationLoader`.

These tags can be specified even if the configuration item class does not have a `MyAssemblyConfig` or `MyClasspathConfig` property. If you use these tags, you need to make sure that the top-level configuration item tag is a fully specified class name or you need to ensure that the prefix for the top-level configuration item tag is specified when creating the generic loader.

Tag Name	Description
<SomeCustomConfig/>	(Optional) You use this tag to nest the <MyAssemblyConfig/> and the <MyClasspathConfig/> tags.
<MyAssemblyConfig/>	<p>(Optional) Specifies the assembly to be loaded before attempting to load the class for the parent tag.</p> <p>Nested tags:</p> <ul style="list-style-type: none"> ■ (Required) <FullName/> The full name of the assembly to be loaded before loading the class specified in the <JavaName/> tag of the factory tag. ■ (Optional) <InGAC/> A boolean value of true that specifies that the assembly is registered in the Global Assembly Cache (GAC). If the tag is not specified, the default is false. ■ (Optional) <HintPath/> The hint path for the location of the assembly. Use this tag to load the assembly when the a configuration is loaded using the Blaze Advisor IDE, or an Advisor utility such as the NdRomAdminUtil or the NdAdvisorProjectCompiler. The specified path can be either absolute or relative (recommended). <p>When a configuration is obtained from a file, the path is interpreted as relative to the current working directory of the application.</p>
<MyClasspathConfig/>	<p>(Optional) Specifies the classpath configuration needed to load the class for the parent tag <SomeCustomConfig>. It is possible to write a configuration file that supports both .NET and Java. You can have the <MyAssemblyConfig/> used for the .NET version and the <MyClasspathConfig/> used for the Java version, present in the same configuration file.</p> <p>Nested tag:</p> <p><ClasspathEntry/> Specifies the classpath entry needed to load the class specified in the <JavaName/> tag of the factory tag. The classpath can be a directory or the location of a jar file. The specified path can be either absolute or relative (recommended).</p> <p>When a configuration is obtained from a file, the path is interpreted as relative to the current working directory of the application.</p>

Example:

If you have a custom class or assembly and you want to be able to load this class or assembly from an application that does not have the class on its classpath or is not linked to the assembly, you can write a self-contained configuration file, similar to the one below:

```

<SomeCustomConfig>
    <MyAssemblyConfig>
        <FullName> customClassAssembly.dll, Version=1.0.0.0, Culture
neutral,
        PublicKeyToken=8457928adjs1kd</FullName>

```

```

<HintPath>.../examples/classes/dotNET/customClassAssembly.dll</
HintPath>
    <InGAC> false </InGAC>
    </MyAssemblyConfig>
    <MyClasspathConfig>
        <ClasspathEntry>.../examples/classes/java </ClasspathEntry>
    </MyClasspathConfig>
</SomeCustomConfig>

```

Embedding a Digital Signature Into a Configuration

After you have edited the repository configuration file, `com.blazesoftware.repository_config.cfg` in your file repository (versioned with a shared workspace) or your workspace (versioned with private or local workspaces) with the tags for the IP protection framework, you need to run the `NdRomAdminUtil` utility to embed a *digital signature* into the configuration.

The privileged user can view the contents of all repository entities and can set or remove protections on those repository entities. During the authentication process, the digital signature is used to verify whether or not the user requesting access to the repository resources is a privileged user by verifying that the security configuration has not been altered since it was signed. If a user is not privileged they may still have access to repository resources, however, they will not be able to view the contents of protected repository entities. Although the non-privileged user cannot view the contents of protected resources, they will be able to compile and execute projects using those resources.

When you run the `NdRomAdminUtil` utility to embed a digital signature, a set of `<Signature/>` tags is added into the configuration before the `<SecurityConfig/>` end tag. If your configuration also contains the `<RepositoryAuthorizationManagerFactory/>` tags, a digital signature is also embedded for the authorization service between a set of `<AuthorizationManagerFactorySignature/>` tags.

If you try to run the utility with the signing command before adding the tags for the Authentication service, you will get an `NdRepositoryException` exception with the message, "The repository does not include a security configuration".

In the batch file or from the command prompt, you reference the path to these files as argument values. See "[Running the NdRomAdminUtil Utility](#)" on page 166.

Command, Argument, or Option	Description
signRepositoryConfig	<p>Checks a repository configuration file for <code><SecurityConfig/></code> tags and if the tags exist, embeds a digital signature for the IP protection framework into the configuration.</p> <p>If the <code><RepositoryAuthorizationManagerFactory/></code> tags exist in the repository configuration embeds a digital signature for the authorization service into the configuration file.</p>

Command, Argument, or Option	Description
-repositoryConfiguration	The path to the com.blazesoftware.repository_config.cfg file in the repository directory.
-verbose	(Optional) Prints out all details messages into the command console.

To run the utility to embed a digital signature into a repository configuration

- 1 If you have not already done so, open a command prompt at <ADVISOR_HOME>
- 2 Set the Blaze Advisor environment using setenv.
- 3 Run the NdRomAdminUtil utility from a command prompt and enter the command and arguments directory in the console or by using a batch file.

This is an example of running the NdRomAdminUtil utility for a versioned repository to add a digital signature using the signRepositoryConfig command:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil  
signRepositoryConfig -repositoryConfiguration "C:\Blaze\Advisor\BVS  
Repo\com.blazesoftware.repository_config.cfg" -verbose
```

- 4 When you have successfully added a digital signature, you see the following message in your command prompt:
`Command <signRepositoryConfig> is executed successfully`

When the signatures are embedded, the contents of the com.blazesoftware.repository_config.cfg look similar to this example:

```
<?xml version="1.0" encoding="UTF-8" ?>  
<RepositoryConfig>  
    <! AtticRepository tags are auto-generated for most repository types and its value of  
    true is only relevant for a BVS shared repository -->  
    <AtticRepository> false </AtticRepository>  
...  
    <AuthorizationManagerFactorySignature>tkeUHWX9L0fd5+qHSSAuTlieNCvoxEN2VwYhJvHsq3  
    xyQQ6HZecSusFCcU0ZTgzmZTqppNZ+2Mw43rk7iaq3sYfK1DHYyMgyG+CnZyLYHvjmhuxAT1  
    doNvLAX7thYapu5eU0U1U0n8iL9nuABiMvztMVznBIz3wx0txmTwPHZ6X9baaGkuBL+UoYI  
    ZfRZDuGpeNxJuQA78+0Dk06TYK3ed4D7+AeMSjqOUXuTGILAVxZ/uWk6E0i0K0eN+OnuzoF  
    VqpjvDy9/Kto59LWhB9NoBBLdCxQ1/I3wqXS6/VwFkwOy+D4ILe/  
    bDsz8GhlQeDDtyRCAhKkAspQw0u1LEQQ==  
    </AuthorizationManagerFactorySignature>  
    <RepositoryAuthorizationManagerFactory>  
        <JavaName>com.blazesoftware.template.repository.security.NdDefSecureProm  
            AuthorizationManager </JavaName>  
        </RepositoryAuthorizationManagerFactory>  
  
<SecurityConfig>  
    <AuthenticationServiceConnection>  
        <Factory> com.blazesoftware.repository.security.NdDefaultAuthentication  
            ServiceConnection </Factory>  
        <PrivilegedIdentityToken> root </PrivilegedIdentityToken>  
    </AuthenticationServiceConnection>  
  
<AuthenticationServiceFactory>  
    <JavaName> examples.ExampleAuthenticationService </JavaName>
```

```

</AuthenticationServiceFactory>
<EncryptedContentConverterFactory>
    <JavaName> examples.ExampleEncryptedContentConverter </JavaName>
</EncryptedContentConverterFactory>

<Signature>x4+r5NdHdDUT6uN8NFB1NEYoKGt1DLc0iQXbrfdhGL+OL+0ZeeHIzRveUj
7x41johowAotiuUewD9fTEXJ3VFA5xwxz98xmtlRo+T5cA9qHiZjGJ0dkqHD8z
pQZB0NIdgdPmBpUADPBqUtGzZ4S8PEMHB6E9wpgk4trFop+1PLxk42woXsBRMh
tnOhHpGW6XjRkWlCExdJ0j4pL09Jzu7YFU0g6h4pVvy0HMSy2g821IY2XdWt7S
0XoeYkvntTs94P9ZxuNL8BGR2m43LSxf4UzqBTMsCAw9ZCfWbAy0YuXRwihcAno0hC3j
bGAJQYImntivpxOSixv5AkWUjrQ==
</Signature>
</SecurityConfig>
...
</RepositoryConfig>

```

Verifying a Digital Signature

You can run the `NdRomAdminUtil` utility to verify the signatures in a repository or workspace configuration `com_blaresoft_repository_config.cfg` file.

In the batch file or from the command prompt, you reference the path to these files as argument values. See ["Running the NdRomAdminUtil Utility" on page 166](#).

Command, Argument, or Option	Description
<code>verifyRepositoryConfig</code>	Checks a repository configuration file for <code><Signature/></code> and/or <code><AuthorizationManagerFactorySignature/></code> tags and verifies its/their validity.
<code>-repositoryConfiguration</code>	The path to the <code>com_blaresoft_repository_config.cfg</code> file in the repository directory.
<code>-verbose</code>	(Optional) Prints out all details messages into the command console.

To run the utility to verify digital signatures in a repository configuration

- If you have not already done so, open a command prompt at `<ADVISOR_HOME>`
- Set the Blaze Advisor environment using `setenv`.
- Run the `NdRomAdminUtil` utility from a command prompt and enter the command and arguments directory in the console or by using a batch file.

This is an example of running the `NdRomAdminUtil` utility for a versioned repository to add a digital signature using the `verifyRepositoryConfig` command:

```
java com.blazsoft.template.repository.admin.NdRomAdminUtil
verifyRepositoryConfig -repositoryConfiguration "C:\Blaze\Advisor\BVS
Repo\com_blaresoft_repository_config.cfg" -verbose
```

- When you have successfully added a digital signature, you see the following message in your command prompt:
`Signatures are good.`
`Command <verifyRepositoryConfig> is executed successfully`

CHAPTER 5

Repository Administration Reference

This chapter describes how to write the connection and configuration files and use with the `NdRomAdminUtil` administrative utility in the FICO® Blaze Advisor® decision rules management system. You can use the utility to perform many tasks including those for importing a workspace or project, exporting a workspace or project, and updating system folder contents. This utility can also used to create a repository or workspace based on a given Blaze Advisor repository type.

This chapter includes information about:

- ["The NdRomAdminUtil Utility" on page 161](#)
- ["The NdRomAdminUtil Utility Commands" on page 162](#)
- ["Running the NdRomAdminUtil Utility" on page 166](#)
- ["About Writing Files for Creating a Blaze Advisor Repository or Workspace Using the Utility" on page 168](#)
- ["Writing Connection Configuration Files" on page 169](#)
- ["Writing a Repository Configuration File" on page 191](#)

The NdRomAdminUtil Utility

The `com.blazesoft.template.repository.util.NdRomAdminUtil` administrative utility uses an default class that implements the `NdRomAdmin` interface. See the Help > API Reference > Innovator (Frames version or Non-frames version).

The utility is used to perform many administrative tasks in a command prompt including, creating a repository, creating a workspace, importing or exporting a workspace or project, synchronize the system folders, or converting a workspace using a one repository connection to use another repository connection type.

To use the utility to perform these tasks you need to write XML configuration files for the repository or workspace type you want to use. See ["Writing Connection Configuration Files" on page 169](#).

After you have written your files, you can run the `NdRomAdminUtil` utility and enter the appropriate arguments, argument values, options, and option values. See ["Running the NdRomAdminUtil Utility" on page 166](#).

For information about the tasks you can perform using the utility, see ["Connection Parameter Sub-arguments" on page 67](#).

The NdRomAdminUtil Utility Commands

The NdRomAdminUtil utility is used to perform many administrative tasks including synchronizing the system folder, creating repositories and workspaces, and tasks related to working with files tracked using a versioning service. The following are the utility commands currently available. You can also get a similar list by using the command-line Help for the utility, see “[Obtaining Help from the Command Prompt](#)” on page 165.

Command	Description
createRepository	Create a repository. See “ Creating a Blaze Advisor Repository Using the Utility ” on page 33.
createWorkspace	Create a private workspace for a BVS repository or a local workspace for a CVS repository. See “ Creating a Blaze Advisor Workspace Using the Utility ” on page 228. Note “ Creating a Blaze Advisor Repository Using the Utility ” on page 33.
removeRepository	Remove a repository. See “ Removing a Blaze Advisor Repository Using the Utility ” on page 37.
removeWorkspace	Remove a workspace. See “ Removing a Blaze Advisor Repository Using the Utility ” on page 37.
importWorkspace	Import a workspace from a .zip file. See “ Importing a Workspace from a Zip File Using the Utility ” on page 74.
exportWorkspace	Export a workspace to a .zip file. See “ Exporting a Workspace to a Zip File Using the Utility ” on page 74.
importProject	Imports a project from a source workspace to a .zip file. See “ Importing a Project from a Zip File Using the Utility ” on page 68.
exportProject	Exports a project from your current workspace to a .zip file. See “ Exporting a Project to a Zip File Using the Utility ” on page 71.
exportImportWorkspace	Export a workspace into another workspace. This command can be used to convert a workspace to use another default repository type. See “ Exporting and Importing a Workspace Using the Utility ” on page 75.
exportImportProject	Export a project into another workspace. See “ Exporting and Importing a Project ” on page 72.

Command	Description
releaseProject	Release a project to another location in the same repository. See “ Releasing a Project Using the Utility ” on page 76.
showReleaseManager	Show a release manager class. See “ Showing a Release Manager ” on page 78.
setReleaseManager	Set a release manager class. See “ Setting a Release Manager ” on page 78.
publishProject	Publish a project to another workspace. See “ Publishing a Project Using the Utility ” on page 79.
updatePublishedProject	Update a published project. See “ Updating a Published Project Using the Utility ” on page 81.
replacePublishedProject	Replace a published project. See “ Replacing a Published Project Using the Utility ” on page 82.
listQueries	Lists configured Java queries. See “ Listing Queries ” on page 99.
addQuery	Add a Java query. See “ Adding a Query ” on page 97.
removeQuery	Remove a Java query. See “ Removing a Query ” on page 98.
status	Show the status of a versioned entry. See “ Checking the Status of Workspace Files Using the Utility ” on page 90.
history	Show the history of a versioned entry. See “ Viewing Item History Using the Utility ” on page 89.
update	Update the workspace or a specific entry location. See “ Updating a Private or Local Workspace Using the Utility ” on page 91.
checkout	Check out a entry. See “ Checking Out One or More Items Using the Utility ” on page 85.
cancelCheckout	Cancel the check out of an entry. See “ Canceling the Check Out for One or More Items Using the Utility ” on page 86.
checkin	Check in an entry. See “ Checking In New or Existing Items Using the Utility ” on page 84.
label	Add a label to a workspace entry. See “ Adding Labels to a Workspace Entry ” on page 92.
getLabels	Get all labels for a workspace entry. See “ Retrieving Labels for a Workspace Entry ” on page 93.

Command	Description
deleteLabel	Delete a label from a workspace entry. See “ Deleting a Label from a Workspace Entry ” on page 94.
delete	Logically delete an entry. See “ Deleting a Versioned Item Using the Utility ” on page 87.
restore	Restore a versioned workspace entry. See “ Restoring Logically Deleted Items ” on page 88.
listLocks	Lists the entries that are locked. See “ Obtaining a List of Locked Files ” on page 95.
releaseLocksByUser	Releases entries locked by a specific user. See “ Unlocking Files by User ” on page 96.
changeConfig	Changes the workspace ROM configuration. See “ Changing an Existing Repository Configuration ” on page 144.
commitConfig	Commits workspace ROM configuration changes to the repository. See “ Committing a Configuration Change from a Workspace to a Repository ” on page 150.
updateConfig	Updates the workspace ROM configuration from the repository. See “ Updating a Configuration Change from a Repository to a Workspace ” on page 151.
synchronizeSystemFolder	Synchronizes the workspace system folder with the system folder in the Admin Repository in <ADVISOR_HOME>/lib. You can use synchronizeSystemFolder when you want to revert the contents of your system folder to the default contents in the system folder in the Admin Repository. See “ Synchronizing the system Folder ” on page 137.
applyFix	Update the system folder in a given workspace based on a configuration file supplied by Blaze Advisor or by adding the last check in attributes. See “ Applying Fixes ” on page 99.

Command	Description
signRepositoryConfig	<p>Checks the repository or workspace configuration to see if the <SecurityConfig/> tags exist and adds a digital signature nested within <Signature/> tags to prevent non-privileged users from viewing the contents of repository entities.</p> <p>If the repository configuration contains <RepositoryAuthorizationManagerFactory/> tags, this command also adds a digital signature nested within <AuthorizationManagerFactorySignature/> tags.</p> <p>See “Embedding a Digital Signature Into a Configuration” on page 157.</p> <p>For information about adding an authentication service, see “Adding an Authentication Service for IP Protection” on page 196.</p>
verifyRepositoryConfig	<p>Verify that the repository or workspace configuration file contains digital signatures.</p> <p>See “Verifying a Digital Signature” on page 159.</p>
encrypt	<p>Encrypt the supplied password.</p> <p>See “Using NdRomAdminUtil to Encrypt a Password” on page 48.</p>
Updating Business Object Models	<p>See “BOM Admin APIs” in <i>APIDevelopersGuide.pdf</i>.</p>

Obtaining Help from the Command Prompt

You can view the command-line Help for the `NdRomAdminUtil` utility by using -H option and one of the following terms:

- **Commands**

Use this term to see a list of all the commands you can use with the `NdRomAdminUtil` utility.

Example:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil -H Commands
```

- **Command Name**

Enter one of the command names you see displayed in the list of commands to obtain specific usage information. The command name must be entered exactly as you see it in the list of commands.

Example:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil -H checkout
```

- **Arguments**

Use this term to see a list of arguments you can use with the utility.

Example:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil -H Arguments
```

- **Options**

Use this term to see a list of options you can use with the utility.

Example:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil -H Options
```

Running the NdRomAdminUtil Utility

You can run the **NdRomAdminUtil** utility from a command prompt and enter the command, arguments, and values in the console or by using a batch file. In most cases the argument values are pathnames to configuration files. The information you enter in the command console or the batch file is similar. If you enter the information in the command console or use a batch file, you need to set the Blaze Advisor environment variables before running the utility.

Example of command prompt input for creating a File repository using pathnames:

```
C:\Blaze\Advisor\bin> setenv  
C:\Blaze\Advisor\bin> java  
com.blazesoftware.template.repository.admin.NdRomAdminUtil createRepository -  
repositoryConnection C:\Repo_Connections\ConnectionFiles\DefaultRepo.cfg -  
systemConnection C:\Repo_Connections\ConnectionFiles\sysAdmin.cfg -  
configuration C:\Blaze\Advisor\lib\NoVerGeneric.cfg -verbose
```

Example of a batch file (.bat) to create a File repository using pathnames:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil  
createRepository -repositoryConnection  
“C:\Repo_Connections\ConnectionFiles\DefaultRepo.cfg” -systemConnection  
“C:\Repo_Connections\ConnectionFiles\sysAdmin.cfg” -configuration  
“C:\Blaze\Advisor\lib\NoVerGeneric.cfg” -verbose
```



Note Use double quotes when using pathnames with spaces.

The **NdRomAdminUtil** utility displays provides a list of Commands, Arguments, and Options. It also provides usage information for individual commands. See “[Obtaining Help from the Command Prompt](#)” on page 165.

Entering a Symbolic Name or a Pathname as an Argument Value

You can use either a *pathname* or a *symbolic name* as an argument value when running the **NdRomAdminUtil** utility either through a command prompt or a batch file. These argument values are used to provide connection configuration information when you are performing administrative tasks using the utility. Of the two types of argument values, pathnames can be used for most utility tasks requiring connection configuration information. Although symbolic names are not as flexible, they can be used in cases where a repository connection has already been created in the Blaze Advisor IDE.

- “[Entering a Pathname](#)” on page 167
- “[Entering a Symbolic Name](#)” on page 167

Entering a Pathname

You can use a pathname as an argument value when running the `NdRomAdminUtil` utility to perform administrative tasks. The pathname is usually the directory location where the connection configuration or repository configuration file is located on your local machine.

Example batch file for creating a File repository using pathnames:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil
createRepository
-repositoryConnection "C:/Repo_Connections/ConnectionFiles/DefaultRepo.cfg"
-systemConnection "C:/Repo_Connections/ConnectionFiles/sysAdmin.cfg" -
-configuration "C:/Blaze/Advisor/lib/NoVerGeneric.cfg" -verbose
```

Entering a Pathname for a Directory Location

If you need to add an argument value for a directory, project, or entry location in a workspace when you run the `NdRomAdminUtil` utility, the location is written as a pathname relative to the root directory.

Example

```
-directoryLocation /"Project1 Folder"
```



Note Use double quotes when using pathnames with spaces.

Entering a Symbolic Name

You can use a symbolic name as an argument value when using the `NdRomAdminUtil` utility to perform administrative tasks for existing repositories and workspaces. The administrative tasks you can perform include, changing a repository configuration, importing or exporting projects and workspaces, or synchronizing the system folder. For more information about these tasks, see:

- “[Changing an Existing Repository Configuration](#)” on page 144
- “[Synchronizing the system Folder](#)” on page 137
- “[Creating and Using Blaze Advisor Rule Projects](#)” in *DevelopingRuleProjects.pdf*
- “[About Creating and Managing Project Content](#)” in *DevelopingRuleProjects.pdf*

The symbolic name is the name of the connection instance file that is created when you use the Manage Connections wizard in Blaze Advisor to create a repository connection. The connection instance file is stored in the Admin Repository. If you are using a non-versioned repository or a versioned repository with a shared workspace where the repository and workspace are considered to be located in the same physical location, the connection instance file contains a reference to the repository directory. If you are using a versioned repository with a local or private workspace, the connection instance file contains a reference to both the local folder (workspace) and the repository directory. With the exception of using a symbolic name for the Admin Repository, you cannot use symbolic names to create a repository or a workspace.

To use a symbolic name when running the NdRomAdminUtil utility

- 1 Create a new connection to the repository using the Manage Connections wizard in Blaze Advisor. See “Connecting to a Blaze Advisor Repository” in *DevelopingRuleProjects.pdf*.

The name you enter in the repository connection name field on the first page of the wizard is the name of the connection instance file and the name you will use for the symbolic name.

- If you have created a new connection to a repository, you can use the symbolic name when you need to use the `-workspaceConnection` argument with the utility.
- If you have created a connection to the Admin Repository you use the symbolic name when you need to use the `-systemConnection` argument with the utility.

- 2 To use a symbolic name when you run the utility, you need to set the `-DADVISOR_HOME` system property with the location of your Blaze Advisor installation.

You set the `-DADVISOR_HOME` system property option and enter the pathname to your Blaze Advisor installation so that the `NdRomAdminUtil` utility looks in the correct location to find the connection instance files referenced by the symbolic name in the Admin Repository.

Example batch file using a symbolic name for a workspace connection instance file, “FileWksp” that is located in the Admin Repository:

```
java -DADVISOR_HOME="c:\Blaze\Advisor"
com.blazesoft.template.repository.admin.NdRomAdminUtil releaseProject -
workspaceConnection "FileWksp" -projectLocation "/Project_A Folder/
Project_A" -directoryLocation "/Rule Services/Project_A Release 1" -m "New
released project" -verbose
```



Note Use double quotes when using symbolic names with spaces.

About Writing Files for Creating a Blaze Advisor Repository or Workspace Using the Utility

You can use the `NdRomAdminUtil` command-line utility to create a Blaze Advisor repository. To create a non-versioned repository using this utility, you use the `createRepository` command and reference three XML files. You write files for the type of repository connection, the type of repository configuration, and the connection to the system Folder. In order for the repository creation to be successful, the repository connection configuration file and repository configuration file must be for the same repository type. See “[Writing Connection Configuration Files](#)” on page 169.

You use the path name of these files as argument values when you run the utility to create a repository. See “[Running the NdRomAdminUtil Utility](#)” on page 166.

To create a Blaze Advisor workspace for a versioned repository using this utility, after you create the repository, you use the `createWorkspace` command and reference two XML

files. You write files for the workspace connection configuration and you reuse the repository connection configuration file you wrote to create the repository.

You need to write the following files for creating a Blaze Advisor repository using the `NdRomAdminUtil` utility:

- The Admin Repository connection configuration file

The connection configuration file for the Admin Repository that is installed with Blaze Advisor is required so that a copy of the system Folder is added to the repository. The system Folder contains the templates and instances for queries, management properties, verification, and filtering. See “[Repository Connection Configuration File Examples](#)” on page 171.

Alternatively, you can use a symbolic name for the Admin Repository when you run the utility. See “[Entering a Symbolic Name or a Pathname as an Argument Value](#)” on page 166.

- The repository connection configuration file

The connection configuration file for the repository includes information such as the physical location of the repository and any other connection parameters required by the repository type. See “[Repository Connection Configuration File Examples](#)” on page 171.

- The repository configuration file

The configuration file for the repository type includes information any services or workspaces used by the repository. See “[Writing a Repository Configuration File](#)” on page 191.

- (Optional) The workspace connection configuration file for versioned repositories

The connection configuration to the workspace includes the path to the repository directory and the Repository Name. See “[Writing a Connection Configuration File for a Private or a Local Workspace](#)” on page 185.

- Writing a batch file to create a repository Using the utility

See “[Running the NdRomAdminUtil Utility](#)” on page 166.

Writing Connection Configuration Files

You write connection configuration XML files (.cfg) when you want to use the `NdRomAdminUtil` utility to perform administrative tasks. Most of the commands you can use with the utility require at least one connection configuration file. For example, you use a repository configuration file, a connection configuration file for the Admin Repository, and a connection configuration for a repository when you want to create a repository.

For information on writing a repository configuration file, see “[Writing a Repository Configuration File](#)” on page 191.

Note To successfully create a repository, the configuration and connection type must be the same.

This topic contains the following sections:

- “[Repository Connection Configuration File Examples](#)” on page 171
- “[Writing a Connection Configuration File for a Private or a Local Workspace](#)” on page 185

The Blaze Advisor installation provides sample files that you can use to base your connection and configuration files. See “[Sample Connection and Configuration Files Installed With Blaze Advisor](#)” on page 170.

Sample Connection and Configuration Files Installed With Blaze Advisor

Sample connection configuration and repository configuration files are provided in the \lib directory of your Blaze Advisor installation. You can open a file in a text editor, copy and paste the contents, edit the values, and save your changes using another file name so you can use these files with the NdRomAdminUtil utility.

The following samples are provided:

File name	Description
adminConfig.cfg	Sample of an Admin Repository connection configuration file.
romConfig.cfg	Sample of a repository configuration file. This file is configured for a repository with BVS versioning. You edit this file when you want to change the configuration of an existing repository. See “ Changing an Existing Repository Configuration ” on page 144. For a sample repository configuration file for a non-versioned repository, see “ Repository Configuration With No Services ” on page 202. For information about the file contents, see “ Writing a Repository Configuration File ” on page 191.
fileBVSRepository.cfg	Sample of a File BVS Repository connection configuration file. Note You can use this file as a basis for another repository type such as a Database type. To write a connection configuration file for the Admin Repository. See “ Repository Connection Configuration File Examples ” on page 171. For information about the file contents, see “ Repository Connection Configuration File Examples ” on page 171.

File name	Description
fileCVSRepository.cfg	Sample of a File CVS Repository connection configuration file. For information about the file contents, see "File Repository with CVS Versioning" on page 177 .
fileWorkspace.cfg	Sample of a File workspace connection configuration file. For information about the file contents, see "File Workspace" on page 186 .
jdbcWorkspace.cfg	Sample of a Database workspace connection configuration file. For information about the file contents, see "Database Workspace" on page 187 .
mongoNVRep.cfg	Sample of a MongoDB non-versioned repository connection configuration file. For information bout the file contents, see "Creating a MongoDB Repository (no versioning) Using the NdRomAdminUtil Utility" on page 339 .
mongoBVSRepository.cfg	Sample of a MongoDB BVS repository connection configuration file. For information about the file contents, see "Creating a MongoDB Repository (BVS versioning - private) Using the NdRomAdminUtil utility" on page 350 .
mongoWorkspace.cfg	Sample of a MongoDB Workspace connection configuration file. For information about the file contents, see "Creating a Workspace for a MongoDB Repository (versioning - private) Using the Blaze Advisor IDE" on page 365 .

Repository Connection Configuration File Examples

You write an XML file (.cfg) with the connection parameters for the type of repository you want to create using the `NdRomAdminUtil` utility. The connection configuration file for this repository contains information including the physical location of the repository and user logon parameters that are specific for the repository type. You write this file using a text editor.

Each repository type supported by Blaze Advisor uses a set of connection parameters that are specific to its repository type.

- ["File Repository" on page 172](#)
- ["Database Repository" on page 173](#)
- ["Repository with BVS Versioning" on page 175](#)
- ["File Repository with CVS Versioning" on page 177](#)
- ["File Repository \(Subversion versioning\)" on page 179](#)
- ["File Repository \(ClearCase versioning\)" on page 181](#)

If you want to create a new workspace, see ["Writing a Connection Configuration File for a Private or a Local Workspace" on page 185](#).

If you want to create a workspace for a MongoDB repository, see “[Creating or Removing a MongoDB or File Workspace](#)” on page 365.

The connection file you write must match the repository configuration you will be using. See “[Writing a Repository Configuration File](#)” on page 191.

File Repository

The repository configuration file (.cfg) you use when you run the `NdRomAdminUtil` utility determines the repository type and the available services. See “[Writing a Repository Configuration File](#)” on page 191.

The connection configuration file (.cfg) for a File repository should contain the following structure and information:

 **Note** For reasons of brevity, a pair of element tags will be represented by the following notation, <example>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<RepositoryConnection/>	The RepositoryConnection tag is used when you need to create a repository connection. All other tags in this table are nested within this tag.
<Factory/>	The Factory tag is used when you want to create an instance of a class. To create an instance of a connection for a File repository, you use the fully qualified class name: <code>com.blazesoft.repository.file.NdFileRepositoryConnection</code> .
<Name/>	The Name tag is used to specify the connection name listed in the Manage Connections wizard.
<RepositoryFolder/>	The RepositoryFolder tag is used to specify the physical location of the repository.
<User/>	The User tag is used to specify the name of the user.
<Password/>	(Optional) The Password tag is used to specify the password if necessary.
<RepositoryName/>	The RepositoryName tag is used to specify the name of the root directory. This value is displayed at the root of the repository structure when you are connected to the repository in Blaze Advisor.

 **Note** To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<RepositoryConnection>
    <Factory>
```

```

com.blazesoftware.repository.file.NdFileRepositoryConnection
</Factory>
<RepositoryName>File Repository</RepositoryName>
<RepositoryFolder>C:\Blaze\Advisor\lib\File Repository345</
RepositoryFolder>
<RepositoryName> File Repository345</RepositoryName>
</RepositoryConnection>

```

Database Repository

The repository configuration file (.cfg) you use when you run the `NdRomAdminUtil` utility determines the repository type and the available services. See ["Writing a Repository Configuration File" on page 191](#).

The connection configuration file (.cfg) for a Database repository should contain the following structure and information:



Note For reasons of brevity, a pair of element tags will be represented by the following notation, <example/>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<RepositoryConnection/>	The RepositoryConnection tag is used when you need to create a repository connection. All other tags in this table are nested within this tag.
<Factory/>	The Factory tag is used when you want to create an instance of a class. To create an instance of a connection for a File repository, you use the fully qualified class name: <code>com.blazesoftware.repository.jdbc.NdJdbcRepository Connection</code>
<PersistCredentials/>	(Optional) Used to save the username and password to the connection instance file in the Admin Repository. If you are using a BVS repository with private workspaces, the value is applied for both the repository connection and the workspace connection. Possible values: <ul style="list-style-type: none"> ■ false -- do not store connection credentials (default) If you choose not to store the connection information, each time you connect to the repository in the Blaze Advisor IDE, you need to manually enter your user name and password. ■ true -- store the connection credentials See "File Repository (Subversion versioning)" on page 179.
<JdbcUrl/>	The URL for the database server. Example: <code>jdbc:hsqldb:hsq://localhost:9001</code>

Tag	Description
<Name/>	The Name tag is used to specify the connection name listed in the Manage Connections wizard.
<JdbcDriver/>	The JdbcDriver tag is used to specify the database driver class. This class must be on your classpath. Example: org.hsqldb.jdbcDriver
<JdbcTable/>	The JdbcTable tag is used to specify the name of the table you created in the database to house the repository. See “ Creating a Database Table ” on page 287.
<JdbcAccessor/>	The JdbcAccessor tag is used to specify the Jdbc accessor class. The default class is: com.blazesoft.repository.jdbc.NdDefaultJdbcRepositoryAccessor Note If you are using Oracle version earlier than 11g, you need to use the following class: com.blazesoft.repository.jdbc.NdOracleJdbcRepositoryAccessor. Otherwise, you can use the default class, com.blazesoft.repository.jdbc.NdDefaultJdbcRepositoryAccessor.
<User/>	The User tag is used to specify the user name for the database server connection. For example for the HSQL database server, the user name is “sa”.
<Password/>	The Password tag is used to specify the password for the database server connection. If no password is necessary you can omit this tag from the connection file.
<RepositoryName/>	The.RepositoryName tag is used to specify the name of the repository. This value is displayed at the root directory when you are connected to the repository in the Blaze Advisor IDE.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
?xml version="1.0" encoding="UTF-8" ?
<RepositoryConnection>
    <Factory>
        com.blazesoft.repository.jdbc.NdJdbcRepositoryConnection
    </Factory>
    <PersistCredentials> true </PersistCredentials>
    <JdbcUrl>jdbc:hsqldb:hsq://localhost:9001 </JdbcUrl>
    <JdbcDriver> org.hsqldb.jdbcDriver </JdbcDriver>
    <JdbcTable> RepoTable10307 </JdbcTable>
    <JdbcAccessor>
```

```

com.blazesoftware.repository.jdbc.NdDefaultJdbcRepositoryAccessor
<JdbcAccessor>
<User>sa </User>
<RepositoryName>DatabaseTestRepo </RepositoryName>
</RepositoryConnection>

```

Repository with BVS Versioning

You can write a connection configuration file (.cfg) for a file or Database repository with BVS versioning. This connection file needs to contain the structure and the values described in the table below.

You can use a shared or a private workspace with a File BVS repository and a private workspace with a Database repository. If you use a private workspace, you need to write a separate connection configuration file and run the utility a second time to create the workspace. See ["Writing a Connection Configuration File for a Private or a Local Workspace" on page 185](#).



Note For reasons of brevity, a pair of element tags will be represented by the following notation, <example/>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<VersioningRepositoryConnection/>	The VersioningRepositoryConnection tag is used when you need to create a connection for a repository with versioning. All other tags in this table are nested within this tag.
<Factory/>	The Factory tag is used when you want to create an instance of a connection for BVS versioning, you use the fully qualified class name. For example, com.blazesoftware.repository.generic.version.NdNativeVersioningRepositoryConnection.
<RepositoryConnection/>	The RepositoryConnection tags nest the connection parameter tags for the specific repository type using versioning. The tags and the values you enter for the connection parameters are the same as those you would use when writing a connection file for a repository of that type without versioning: <ul style="list-style-type: none"> ■ "File Repository" on page 172 ■ "Database Repository" on page 173
<User/>	This User tag is used to specify the user name for the connection to the versioned repository and appears after the closing inner RepositoryConnection tag and before the outer RepositoryConnection end tag. You need to use this tag if you have a repository with BVS versioning. If you are using BVS versioning you also use this tag before the VersioningRepositoryConnection end tag.

Tag	Description
<Password/>	This Password tag is used to specify the password for the connection to the versioned repository and appears after the closing RepositoryConnection tag and before the VersioningRepositoryConnection end tag. You need to use this tag if you have a connection to a Database server or use an Authorization service that requires a password. If you are using a password with your connection, you also use this tag before the VersioningRepositoryConnection end tag.
<RepositoryName/>	This Repository Name tag is used to specify the Repository Connection Name. You need to use this tag if you have a File or Database repository with BVS versioning. If you are using BVS versioning you also use this tag before the VersioningRepositoryConnection end tag.
<PersistCredentials/>	(Optional) Used to save the username and password to the connection instance file in the Admin Repository. If you are using a File or Database BVS repository with private workspaces, the value is applied for both the repository connection and the workspace connection. This tag is added after the closing RepositoryConnection tag. Possible values: <ul style="list-style-type: none">■ false -- do not store connection credentials (default) If you choose not to store the connection information, each time you connect to the repository in the Blaze Advisor IDE, you need to manually enter your user name and password.■ true -- store the connection credentials See " Saving User Names and Passwords Using the Utility " on page 183.

After you have written the connection file for your repository with BVS versioning, if you are using a private workspace, you need to write or edit a configuration file for this workspace. See "[Writing a Connection Configuration File for a Private or a Local Workspace](#)" on page 185.

You run the `NdRomAdminUtil` utility twice to create a BVS repository with a private workspace. First you run the utility to create the repository and then you run it a second time to create the workspace. Each time you run the utility, you must use the correct arguments and options for each task. To create a BVS repository with a shared workspace you run the utility once because the repository and the workspace share the same physical location.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example for a File repository with BVS versioning:

```
<?xml version="1.0" encoding="UTF-8" ?>
<VersioningRepositoryConnection>
    <Factory>
        com.blazesoftware.repository.generic.version.NdNativeVersioningRepositoryC
onnection
    </Factory>
    <RepositoryConnection>
        <Factory>
            com.blazesoftware.repository.file.NdFileRepositoryConnection
        </Factory>
        <RepositoryFolder>C:\Repository\ProjectRepos\TestThisRepo3563
        </RepositoryFolder>
        <RepositoryName>Repository_Dir </RepositoryName>
        <User>user1</User>
    </RepositoryConnection>
    <PersistCredentials> true </PersistCredentials>
</VersioningRepositoryConnection>
```

Example for a Database repository with BVS versioning:

```
<?xml version="1.0" encoding="UTF-8" ?>
<VersioningRepositoryConnection>
    <Factory>
        com.blazesoftware.repository.generic.version.NdNativeVersioningRepositoryC
onnection
    </Factory>
    <RepositoryConnection>
        <Factory>
            com.blazesoftware.repository.jdbc.NdJdbcRepositoryConnection
        </Factory>
        <JdbcUrl> jdbc:hsqldb:hsqldb://localhost:9001 </JdbcUrl>
        <JdbcDriver> org.hsqldb.jdbcDriver </JdbcDriver>
        <JdbcTable> BVSRepoTable10307 </JdbcTable>
        <JdbcAccessor>
            com.blazesoftware.repository.jdbc.NdDefaultJdbcRepositoryAccessor
        </JdbcAccessor>
        <User> sa </User>
        <RepositoryName> DatabaseBVS_Dir </RepositoryName>
    </RepositoryConnection>
    <PersistCredentials> true </PersistCredentials>
</VersioningRepositoryConnection>
```

File Repository with CVS Versioning

You can write a connection configuration file (.cfg) for a File repository with CVS versioning that connects to either a CVS or CVSNT server. This connection file needs to contain the structure and the values described in the table below.

You can use a local workspace with a CVS repository. In the repository configuration file, when you specify that you are using a CVS repository, you automatically use a local workspace. You need to write a separate connection configuration file for the local workspace and run the utility a second time. See ["Writing a Connection Configuration File for a Private or a Local Workspace"](#) on page 185.

 **Note** For reasons of brevity, a pair of element tags will be represented by the following notation, <example/>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<Repository Connection/>	Used when you need to create a connection for a repository with versioning. All other tags in this table are nested within this tag.
<Factory/>	Used when you want to create an instance of a class. To create an instance of a connection for CVS versioning, you use the fully qualified class name: com.blazesoft.repository.file.NdFileCVSServer Connection.
<PersistCredentials/>	(Optional) Used to save the username and password to the connection instance file in the Admin Repository. Possible values: <ul style="list-style-type: none">■ false -- do not store connection credentials (default) If you choose not to store the connection information, each time you connect to the repository in the Blaze Advisor IDE, you need to manually enter your user name and password.■ true -- store the connection credentials See “Saving User Names and Passwords Using the Utility” on page 183.
<DisableMultipleCommands/>	Used if you have a CVS server, such as a CVSNT server, that cannot handle multiple commands. By default, the value for the tag is false. If you have a CVSNT server, the value is true. See “ Creating a CVSNT Repository ” on page 248.
<Type/>	Used to specify the connection type. The connection type currently supported by Blaze Advisor is pserver.
<Host/>	Used to specify the name of the host machine. This is the machine that has the CVS server installed.
<Name/>	Used to specify the name of the repository module that houses the CVS repository.
<Port/>	Used to specify the port number. The default port number is 2401.
<Path/>	Used to specify the path to the module.
<RepositoryName/>	The name of the repository directory on your local machine. This value must be the same as the repository module specified in the <Name/> tag.
<User/>	Used to specify the user name for the connection to the CVS server.

Tag	Description
<Password/>	Used to specify the password for the connection to the CVS server.
<Branch/>	(Optional) If branching is used, the name of the branch.

After you have written the connection file for your CVS repository, you need to write or edit a configuration file for a local workspace. See ["Writing a Connection Configuration File for a Private or a Local Workspace" on page 185](#).

You run the `NdRomAdminUtil` utility twice to create a repository with a local workspace. First you run the utility to create the repository and then you run it a second time to create the workspace. Each time you run the utility, you must use the correct arguments and options for each task.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<RepositoryConnection>
  <Factory> com.blazesoftware.repository.file.NdFileCVSServerConnection </
Factory>
  <PersistCredentials> true </PersistCredentials>
  <DisableMultipleCommands> false </DisableMultipleCommands>
  <Type>pserver </Type>
  <Host>boxster </Host>
  <Port>2401 </Port>
  <Path>/projects/k/masters/repository2 </Path>
  <Branch>repo3xbranch</Branch>
  <User>user1 </User>
  <Password>user1 </Password>
  <RepositoryName>CVSRepo100</RepositoryName>
</RepositoryConnection>
```

File Repository (Subversion versioning)

You can write a connection configuration file (.cfg) for a File repository (Subversion versioning). This connection file needs to contain the structure and the values described in the table below.

You use a local workspace with this repository type. When you create Subversion repository using the utility, you also need to specify the path to the workspace connection configuration file. See ["File Workspace" on page 186](#).



Note For reasons of brevity, a pair of element tags will be represented by the following notation, <example/>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<VersioningRepository Connection/>	Used when you need to create a connection for a Subversion or ClearCase repository. All other tags in this table are nested within this tag.
<Factory/>	Used when you want to create an instance of a class. com.blazesoft.repository.scm.NdScmServer Connection
<PersistCredentials/>	<p>(Optional) Used to save the username and password to the connection instance file in the Admin Repository.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ■ false -- do not store connection credentials (default) If you choose not to store the connection information, each time you connect to the repository in the Blaze Advisor IDE, you need to manually enter your user name and password. ■ true -- store the connection credentials <p>See “Saving User Names and Passwords Using the Utility” on page 183.</p>
<RepositoryVersionSystem AdminFactory/>	Used to create an instance of the system version administrator class.
<JavaName/>	Used to specify the system version administrator class. This class is installed with Blaze Advisor. com.blazesoft.repository.scm.subversion.NdSvn VersionSystemAdmin
<VersioningSystemVersion ManagerFactory/>	Used to create an instance of the system version manager class.
<JavaName/>	Used to specify the default system version manager class. This class installed with Blaze Advisor. com.blazesoft.repository.scm.subversion. NdSvnVersionSystemVersionManager
<VersionManagerClass/>	Used to specify the version manager used. This class is installed with Blaze Advisor. com.blazesoft.repository.scm.subversion.NdSvn WorkspaceVersionManager
<RepositoryName/>	Used to specify the name of the repository you want to create.
<ServerUrl/>	Used to specify the URL of the host machine or file location for the trunk or branch. Creation of branches and merging is not supported.
<User/>	Used to specify the user name for the connection to the Subversion server.
<Password/>	Used to specify the password for the connection to the Subversion server.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
...
<VersioningRepositoryConnection>
<Factory> com.blazesoftware.repository.scm.NdScmServerConnection </Factory>
    <RepositoryVersionSystemAdminFactory>
        <JavaName>com.blazesoftware.repository.scm.subversion.NdSvnVersionSystemAdmin
        </JavaName>
    </RepositoryVersionSystemAdminFactory>
    <VersioningSystemVersionManagerFactory>
        <JavaName>

            com.blazesoftware.repository.scm.subversion.NdSvnVersionSystemVersionManager
            <JavaName>
            </VersioningSystemVersionManagerFactory>
            <VersionManagerClass>com.blazesoftware.repository.scm.subversion.NdSvnWorksp
ace
                VersionManager</VersionManagerClass>
                <User> user1 </User>
                <Password> pswd123 </Password>
                <RepositoryName> repoTest1 </RepositoryName>
                <ServerUrl> http://servername/DevTeam/RuleProj </ServerUrl>
            </VersioningRepositoryConnection>
            ...

```

File Repository (ClearCase versioning)

You can write a connection configuration file (.cfg) for a File repository (ClearCase versioning). This connection file needs to contain the structure and the values described in the table below.

You use a local workspace with this repository type. When you create ClearCase repository using the utility, you also need to specify the path to the workspace connection configuration file. See ["File Workspace" on page 186](#).



Note For reasons of brevity, a pair of element tags will be represented by the following notation, <example/>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<VersioningRepository Connection/>	Used when you need to create a connection for a Subversion or ClearCase repository. All other tags in this table are nested within this tag.
<Factory/>	Used when you want to create an instance of a class. com.blazesoftware.repository.scm.NdScmServer Connection

Tag	Description
<PersistCredentials/>	<p>(Optional) Used to save the username and password to the connection instance file in the Admin Repository.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ■ false -- do not store connection credentials (default) If you choose not to store the connection information, each time you connect to the repository in the Blaze Advisor IDE, you need to manually enter your user name and password. ■ true -- store the connection credentials <p>See “Saving User Names and Passwords Using the Utility” on page 183.</p>
<RepositoryVersionSystemAdminFactory/>	Used to create an instance of the system version administrator class.
<JavaName/>	Used to specify the system version administrator class. This class is installed with Blaze Advisor. com.blazesoftware.repository.scm.wvcm.NdFileWvcmVersionSystemAdmin
<VersioningSystemVersionManagerFactory/>	Used to create an instance of the system version manager class.
<JavaName/>	Used to specify the default system version manager class. This class is installed with Blaze Advisor. com.blazesoftware.repository.scm.wvcm.cc.NdCcVersioningSystemVersionManager
<VersionManagerClass/>	Used to specify the version manager used. This class is installed with Blaze Advisor. com.blazesoftware.repository.scm.wvcm.cc.NdCcVersionManager
<ScmProvider/>	<p>Used to specify the class used for the ClearCase repository type.</p> <p>Example: com.ibm.rational.stp.client.internal.cc.CcOnlyProviderImpl.</p>
<WorkspaceFolder/>	Used to specify the root of the view.
<RepositoryHomeFolder/>	<p>Used to specify the name of the VOB you want to use.</p> <p>Example: /vobs/yourVOB</p>
<RepositoryName/>	Used to specify the name of the repository you want to create.
<ServerUrl/>	Used to specify the URL of the host machine with the Web client.
<ScmBranch>	<p>If you already have a branch, you can specify its location using this tag.</p> <p>Creation of branches and merging is not supported.</p>
<User/>	Used to specify the user name for the connection to the IBM Rational ClearCase server.
<Password/>	Used to specify the password for the connection to the IBM Rational ClearCase server.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>

<VersioningRepositoryConnection>
<Factory> com.blazesoftware.repository.scm.NdScmServerConnection </Factory>
<RepositoryVersionSystemAdminFactory>

<JavaName>com.blazesoftware.repository.scm.wvcm.NdFileWvcmVersionSystemAdmin
</JavaName>
</RepositoryVersionSystemAdminFactory>
<VersioningSystemVersionManagerFactory>

<JavaName>com.blazesoftware.repository.scm.wvcm.cc.NdCcVersioningSystemVersi
on
    Manager</JavaName>
</VersioningSystemVersionManagerFactory>
<VersionManagerClass>com.blazesoftware.repository.scm.wvcm.cc.NdCcVersionMan
ager
</VersionManagerClass>

<ScmProvider>com.ibm.rational.stp.client.internal.cc.CcOnlyProviderImpl
</ScmProvider>
<WorkspaceFolder>C:\cc_view</rootOfView>
<RepositoryHomeFolder>/vobs/myvob</RepositoryHomeFolder>
<User> user1 </User>
<Password> pswd123 </Password>
<RepositoryName> repoTest1 </RepositoryName>
<ServerUrl> http://serverHost01/ccrc </ServerUrl>
</VersioningRepositoryConnection>
```

Saving User Names and Passwords Using the Utility

If you are creating a repository using one of the default types using the `NdRomAdminUtil` utility, you have the option of saving your user name and password to your repository configuration and to a connection instance file in the Admin Repository after you connect to the workspace or repository for the first time in the Blaze Advisor IDE. See ["Managing Repository Configuration and Connection Instances in the Admin Repository"](#) on page 131.

To use this option you need to add the `<PersistCredentials/>` tag and a value of `true` directly after the `<Factory/>` tag for the `<RepositoryConnection/>` in your repository connection configuration file.

- If you set this option to `true` (default), after you connect to the workspace or repository for the first time in the Blaze Advisor IDE you can automatically connect to your workspace or repository by clicking a repository connection name such as those listed in the Manage Connections wizard. You do not have to reenter your workspace credentials each time because the system checks the repository

configuration file storing your user name against the credentials stored in the connection instance file or the credentials you entered.

- If you set this option to `false`, your credentials are not stored locally and you are required to enter the connection credentials manually each time you want to connect.



Note If you are using the utility to create either a ClearCase or Subversion repository, if you set the `<PersistCredentials>` element value to `true`, you see the credentials persisted in the connection instance file but not the repository configuration file.

If you want to use the Blaze Advisor IDE to create your repository and you want to save your workspace and/or repository credentials, see ["Saving User Name and Passwords" on page 43](#).

Allowing Distinct Repository and Workspace Users When Using the Utility

If you are creating non-file BVS repository with private workspaces and want your users to have the ability to enter a user name and password specifically for their workspace and use a common user name and password for the repository server connection, you need to edit the repository configuration with a set of `<Impersonate>` tags with a value of `true` to set the policy option. If you want your users to use their workspace credentials to connect to the repository server connection, you use the default value of `false`.

For example, suppose you want to create a Database BVS repository with private workspaces, you would add the following tags to the file you reference in your -configuration argument:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RepositoryConfig>
    <! AtticRepository tags are auto-generated for most repository types-->
    <!and its value of true is only relevant for a BVS shared repository -->
    <AtticRepository> true </AtticRepository>
    <RepositoryVersionManagerConfig>
        <RepositoryVersionManagerFactory>
            <JavaName>com.blazesoftware.repository.generic.version.
                NdNativeRepositoryVersionManager </JavaName>
            </RepositoryVersionManagerFactory>
        <Impersonate>true </Impersonate>
        <PrivateWorkspaceSupported> true </PrivateWorkspaceSupported>
    </RepositoryVersionManagerConfig>
</RepositoryConfig>
```

You also need inform your users of the policy so that they can create a workspace connection using a user name and password that is not the same as the database server connection. If you are using the Blaze Advisor IDE to create your non-file BVS repository with private workspaces, see ["Allowing Distinct Repository and Workspace Users" on page 49](#).

Writing a Connection Configuration File for a Private or a Local Workspace

If you are creating a File or MongoDB BVS repository with a private file workspace, MongoDB BVS repository with a private MongoDB workspace, or a file CVS, Subversion or Clearcase repository with a local workspace using the NdRomAdminUtil utility, you need to write a connection configuration file (.cfg) for the workspace. You use the pathname to this configuration file when you use the utility to create the workspace. The workspace connection configuration file you write is similar to the connection configuration file you write for a repository. The main difference is that you provide the name and the location of the workspace you want to use in the workspace connection configuration file.



Tip Blaze Advisor provides sample connection and configuration (.cfg) files in the \lib directory of your Blaze Advisor installation. You can open the files in a text editor, edit the values, and save these changes using another file name. See ["Sample Connection and Configuration Files Installed With Blaze Advisor" on page 170](#).

When you use the Blaze Advisor IDE to create a file or MongoDB BVS repository with a private file workspace, a MongoDB BVS repository with a private MongoDB workspace, or a file CVS or SCM repository with a local workspace, a workspace is automatically created for you.

When you use the utility to create a file or MongoDB BVS repository with a private file workspace, a MongoDB BVS repository with a private MongoDB workspace, or a file CVS, SVN or ClearCase repository with a local workspace, you need to run the utility a second time to create the workspace. When you use the utility instead of the Blaze Advisor IDE, you can create a workspace using any of the default Blaze Advisor repository types to connect in the Blaze Advisor IDE, see:

- ["File Workspace" on page 186](#)
- ["Database Workspace" on page 187](#)
- ["MongoDB Workspace" on page 188](#)

The ability to create a workspace of any Blaze Advisor type is useful in cases where you have specific security concerns and do not want to allow sensitive data to be stored in a file system on a local machine. For example, if security is a concern, you may want to create a Database type workspace. If you want to eventually deploy your RMA or rule service to the cloud, you will want to use a MongoDB repository type that uses MongoDB workspaces. Conversely, if you do not know if you will be deploying an RMA or rule service in the cloud, you can use a MongoDB repository type that uses private file workspaces that can easily be converted to a MongoDB workspace when required.

If you are creating a non-file BVS repository with private workspaces, you can allow your users to connect to the workspace using credentials specific to their workspace while at the same time connecting to the server connection. See ["Allowing Distinct Repository and Workspace Users When Using the Utility" on page 184](#).

File Workspace

In most cases you create a private or local file workspace to use with your BVS, CVS or SCM repository. When you create the workspace using the utility, you reference the path to the repository connection configuration file and the workspace connection configuration file.

If you are using the utility to create an SCM repository, the `createRepository` command has a `-workspaceConnection` argument you need to use to specify the path to the workspace connection configuration file. See "["Creating a Blaze Advisor Repository Using the Utility" on page 33](#).

The connection configuration file (`.cfg`) for a file workspace should contain the following structure and information:

 **Note** For reasons of brevity, a pair of element tags will be represented by the following notation, `<example/>`.

Tag	Description
<code><?xml version="1.0" encoding="UTF-8" ?></code>	Processing information that contains the XML version and character-encoding used.
<code><RepositoryConnection/></code>	The <code>RepositoryConnection</code> tag is used when you need to create a repository connection. All other tags in this table are nested within this tag.
<code><Factory/></code>	The <code>Factory</code> tag is used when you want to create an instance of a class. To create an instance of a connection for a File repository, you use the fully qualified class name: <code>com.blazesoft.repository.file.NdFileRepository Connection</code> .
<code><RepositoryFolder/></code>	The <code>RepositoryFolder</code> tag is used to specify the physical location of the workspace on your local machine. If you are creating a File repository (SCM versioning) using a connection to the IBM Rational ClearCase server, this is the path to the view where the VOB is loaded. The workspace is created within the VOB.
<code><User/></code>	The <code>User</code> tag is used to specify the name of the user for the server connection if one exists or the connection to the repository.
<code><Password/></code>	The <code>Password</code> tag is used to specify password for the server connection if one exists or the connection to the repository if applicable.
<code><RepositoryName/></code>	The <code>RepositoryName</code> tag is used to specify the name of the root directory. This value is displayed at the root of the repository structure when you are connected to the repository in Blaze Advisor.

 **Note** To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RepositoryConnection>
  <Factory
    com.blazesoftware.repository.file.NdFileRepositoryConnection
  </Factory>
  <RepositoryFolder> C:\RepoWksp\BVSRepo_Wksp\BVSRepo3563_Dir</
RepositoryFolder>
  <RepositoryName>BVSRepo3563_Dir </RepositoryName>
  <User>user1</User>
  <Password> pswd </Password>
</RepositoryConnection>
```

Database Workspace

If you want to create a connection to a Database workspace, you need to create the workspace using the NdRomAdminUtil utility by implementing a database server and creating a database table, just as you would if you were creating a Database repository. You can create a Database workspace for a BVS File or Database repositories. The connection configuration file you write for the Database workspace would contain information similar to what you include in a Database connection configuration file for a repository:



Important The Database type is not supported for RMA workspaces. By default, the RMA workspaces are of file type.



Note For reasons of brevity, a pair of element tags will be represented by the following notation, <example/>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<RepositoryConnection/>	The RepositoryConnection tag is used when you need to create a repository connection. All other tags in this table are nested within this tag.
<Factory/>	The Factory tag is used when you want to create an instance of a class. To create an instance of a connection for a File repository, you use the fully qualified class name: com.blazesoftware.repository.jdbc.NdJdbcRepository Connection
<JdbcUrl/>	The JdbcURL tag is used to specify the URL for the database server. Example: jdbc:mysql://localhost:3306
<JdbcDriver/>	The JdbcDriver tag is used to specify the database driver class. This class must be on your classpath. Example: org.mysql.jdbc.Driver

Tag	Description
<JdbcTable/>	<p>The JdbcTable tag is used to specify the name of the table you created in the database to house the workspace.</p> <p>See "Creating a Database Table" on page 287.</p>
<JdbcAccessor/>	<p>The JdbcAccessor tag is used to specify the Jdbc accessor class. The default class is: com.blazesoft.repository.jdbc.NdDefaultJdbcRepositoryAccessor</p> <p>Note If you are using Oracle version earlier than 11g, you need to use the following class: com.blazesoft.repository.jdbc.NdOracleJdbcRepositoryAccessor. Otherwise you can use the default class, com.blazesoft.repository.jdbc.NdDefaultJdbcRepositoryAccessor.</p>
<User/>	<p>The User tag is used to specify the user name for the database server connection.</p> <p>For example for the HSQL database server, the user name is "sa".</p>
<Password/>	<p>The Password tag is used to specify the password for the database server connection. If no password is necessary you can omit this tag from the connection file.</p>
<RepositoryName/>	<p>The RepositoryName tag is used to specify the name of the repository. In a connection file for a Database workspace, you would enter the name for the repository you created.</p>



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<RepositoryConnection>
    <Factory>
        com.blazesoft.repository.jdbc.NdJdbcRepositoryConnection
    </Factory>
    <JdbcUrl>jdbc:hsql:hsqldb:hsqldb://localhost:9001 </JdbcUrl>
    <JdbcDriver> org.hsqldb.jdbcDriver </JdbcDriver>
    <JdbcTable> RepoTable10307 </JdbcTable>
    <JdbcAccessor>
        com.blazesoft.repository.jdbc.NdDefaultJdbcRepositoryAccessor
    <JdbcAccessor>
        <User>sa </User>
        <RepositoryName>Repo_Dir </RepositoryName>
    </RepositoryConnection>
```

MongoDB Workspace

You write a workspace connection configuration file for a MongoDB workspace with information including the connection class, the MongoDB URL, the repository connection name, and the repository name. This is similar to the information you would enter in the New Workspace wizard in the Blaze Advisor IDE.

You can create the workspace connection configuration for your MongoDB Repository (versioning - private) using a text editor and save the file with a .cfg extension. The connection configuration file contains the following tag structure and values:



Note For reasons of brevity, a pair of element tags will be represented by the following notation, <example/>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<RepositoryConnection/>	The RepositoryConnection tag is used when you need to create a repository connection. All other connection tags in this table are nested within this tag.
<Factory/>	The Factory tag is used when you want to create an instance of a class. To create an instance of a connection for a MongoDB non-versioned repository, you use the fully qualified class name: com.blazesoftware.repository.mongodb.NdMongoRepositoryConnection.
<MongoDBAuthenticationType/>	The authentication type of the MongoDB instance. Available values are: ■ 0 -- None ■ 1 -- MongoDB See " MongoDB Authentication Type Connection Parameter " on page 315.
<MongoDBURL/>	The URL of the MongoDB instance. See " MongoDB URL Connection Parameter " on page 315. Example: 127.0.0.1:27017
<Name/>	The name is used to specify the connection name listed in the Manage Connections wizard in the Blaze Advisor IDE.
<RepositoryName/>	The name you specify here corresponds to the name used for the workspace database and therefore cannot contain the following characters "/", "\", ".", "*", "<", ">", ".", " ", "?", or double quotes.

Tag	Description
<User/>	<p>The user name.</p> <ul style="list-style-type: none"> ■ If the MongoDB database you are connecting to is running in authenticated mode and you have connected using the MongoDB authentication type, the user name is added to the system.user collection and is used by MongoDB to authenticate the user. The authorization manager also uses the same user name to authenticate the user for access to items in the repository or workspace. ■ If the MongoDB database you are connecting to is running in unauthenticated mode and you have connected using the None authentication type, the MongoDB does not authenticate the user. The authorization manager uses the user name to authenticate the user for access to items in the repository or workspace.
<Password/>	<p>The password.</p> <ul style="list-style-type: none"> ■ If the MongoDB database you are connecting to is running in authenticated mode and you have connected using the MongoDB authentication type, the password is added to the system.user collection and is used by MongoDB to authenticate the user. The authorization manager also uses the same password to authenticate the user for access to items in the repository or workspace. ■ If the MongoDB database you are connecting to is running in unauthenticated mode and you have connected using the None authentication type, the MongoDB does not authenticate the user. The authorization manager uses the password to authenticate the user for access to items in the repository or workspace.
<SharedWorkspace/>	Value is false for this MongoDB repository configuration.
<UseSSLEnabled/>	Value is false. This value is true only if you are using a Secure Sockets Layer (SSL) available only with MongoDB Enterprise Edition. See “ SSL Connections and MongoDB Repositories ” on page 383.
<WorkspaceConnection/>	The value is true. This is a workspace connection.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<RepositoryConnection>
    <Factory>com.blazesoft.repository.mongodb.NdMongo
        RepositoryConnection </Factory>
    <MongoDBAuthenticationType>0</MongoDBAuthenticationType>
```

```

<MongoDBUrl> 127.0.01:27017 </MongoDBUrl>
<Name>MongoWsConnection</Name>
<RepositoryName>MongoWs01</RepositoryName>
<User>user1</User>
<Password>user1</Password>
<SharedWorkspace>false</SharedWorkspace>
<UseSSLEnabled>false</UseSSLEnabled>
<WorkspaceConnection>true</WorkspaceConnection>
</RepositoryConnection>

```

A sample MongoDB repository configuration file `mongoWorkspace.cfg` is located in `<ADVISOR_HOME>/lib`. You may need to edit this file to create a workspace connection configuration. Save a backup copy of this file if you decide to edit its contents.

Writing a Repository Configuration File

You write an XML file (`.cfg`) to configure your repository when you want to use the `NdRomAdminUtil` utility to create a repository or synchronize the system folder. The repository configuration file contains a set of tags and values that reference class names for functionality you want to include in the repository.

When you create the repository, the information you entered in this configuration file or through the Details page of the New Repository wizard is generated as two XML files. See ["Blaze Advisor Repository Contents and File Structure" on page 37](#).

The repository configuration contains similar information that you see on the Details page in the New Repository wizard in the Blaze Advisor IDE. If you can manually edit the repository configuration file to make changes, see ["Editing a Repository Configuration in the Blaze Advisor IDE" on page 39](#).

When you are creating a new repository, you can provide additional services to your repository by the Details page in the New Repository wizard. After you have created the repository, you can manually edit the file to add functionality that is not available through the Details page of the New Repository wizard. For example, you can add a custom class to encrypt your item contents or to filter project items to your repository configuration. If you change the configuration of an existing repository by adding or changing tags or classes, you need to use the `NdRomAdminUtil` utility with the `changeConfig` command. See ["Changing an Existing Repository Configuration" on page 144](#).

There is a sample repository configuration file, `romConfig.cfg`, in the `\lib` directory of your Blaze Advisor installation. You can use this file as a starting point with adding services or use one of the example files provided in the documentation:

- ["Sample Connection and Configuration Files Installed With Blaze Advisor" on page 170](#)
- ["Example Configuration Files" on page 201](#)

The repository configuration file is used with connection configuration files for the repository or workspace where you want to perform the administrative task using the utility. See ["Writing Connection Configuration Files" on page 169](#).

You can add the following services to your repository configuration:

- “Adding a Version Manager” on page 193
- “Adding an Authorization Manager” on page 195
- “Adding an Authentication Service for IP Protection” on page 196
- “Adding a Repository Entry Filter” on page 197
- “Adding an Item Content Converter” on page 198
- “Adding a Query” on page 199
- “Adding a Filter” on page 200
- “Adding Encryption Manager to an Existing Repository” on page 46

If you are using custom classes, you can edit your configuration file to add tags to load those assemblies or classes. See “[Loading Custom Classes](#)” on page 152.

The repository configuration file you write must have the following tags:



Note For reasons of brevity, a pair of element tags will be represented by the following notation, <example>/>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<RomConfig/>	The RomConfig tag is outer tag for the configuration file. Any tags and values added within the RomConfig tags are considered part of the ROM layer and will be displayed in the com.blazesoft_rom_config.cfg in the root directory.
<RomConnectionManagerConfig/>	The RomConnectionManagerConfig contains the following element tags: <ConnectionMode/> <RomConnectionManagerFactory> <JavaName> The default class for the connection manager is com.blazesoft.template.repository.impl.NdDefaultRomConnectionManager. In most cases you will not need to change the information within these tags.

Tag	Description
<RomSchemaManagerConfig/>	<p>The RomSchemaManagerConfig contains the following element tags:</p> <p><RomSchemaManagerFactory> <JavaName></p> <p>The default class for the schema manager is com.blazesoftware.template.repository.impl.NdDefaultRomSchemaManager.</p> <p>In most cases you will not need to change the information within these tags.</p>
<RepositoryConfig/>	<p>The RepositoryConfig tag contains tags and values for classes used in the base repository layer. These tags and values are displayed in the com.blazesoftware.repository_config.cfg that is added to in the root directory when you create a new repository or change the configuration of an existing repository.</p> <p>The following managers use this tag:</p> <ul style="list-style-type: none"> ■ “Adding a Version Manager” on page 193 ■ “Adding an Authorization Manager” on page 195 ■ “Adding an Authentication Service for IP Protection” on page 196 ■ “Adding a Repository Entry Filter” on page 197 ■ “Adding an Item Content Converter” on page 198 ■ “Adding Encryption Manager to an Existing Repository” on page 46

If you are changing the configuration of an existing repository, you need to run the NdRomAdminUtil utility. See [“Changing an Existing Repository Configuration” on page 144](#).

Adding a Version Manager

If you want to add a versioning service to your repository, you write a custom version manager class or use one of the default versioning manager implementation that are available with Blaze Advisor. You add the element tags and values for the version manager within the <RepositoryConfig/> element tags in your repository configuration. When the repository is created, the element tags for the version manager are generated in the com.blazesoftware.repository_config.cfg file located in the root directory.

These configuration files contain a default version manager:

- [“Repository with BVS Versioning” on page 175](#)
- [“File Repository with CVS Versioning” on page 177](#)
- [“Sample Connection and Configuration Files Installed With Blaze Advisor” on page 170](#)

If you want to add versioning services to an existing repository configuration, we recommend that you create a new repository with the default versioning service you

want to use and import the workspace files. See “[Exporting and Importing a Workspace Using the Utility](#)” on page 75.

 **Note** For reasons of brevity, a pair of element tags will be represented by the following notation, <example>.

Tag	Description
<AtticRepository/>	<p>Used when creating a BVS shared repository that requires local revision tracking.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> ■ shared workspaces - true ■ private workspace - false <p>Note This tag is auto-generated and may be included in configuration files that are not for BVS shared repositories. In these cases, this tag and its value are ignored.</p>
<RepositoryVersionManager Config/>	Used to configure the Version Manager.
<RepositoryVersionManager Factory/>	Used to create an instance of the class.
<Impersonate/>	<p>(Optional) Allows distinct repository and workspace users.</p> <p>If you are using a non-file BVS repository with private workspaces, a value of true allows your users to connect to their workspace using a user name and password that differs from the user name and password required to connect to the repository server connection. See “Allowing Distinct Repository and Workspace Users When Using the Utility” on page 184.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> ■ allow the same credentials (default) - false ■ allow different credentials - true
<JavaName/>	<p>Used the following classes for the versioning type:</p> <ul style="list-style-type: none"> ■ For BVS versioning use: com.blazesoft.repository.generic.version.NdNativeRepositoryVersionManager ■ For CVS versioning use: com.blazesoft.repository.file.NdFileCVSWorkspaceVersionManager ■ For a BVS repository you want to archive in a zip file: com.blazesoft.repository.generic.version.NdZipRepositoryVersionManager
<PrivateWorkspaces Supported/>	<p>Used for BVS versioning to specify if you want to use:</p> <ul style="list-style-type: none"> ■ private workspaces - true ■ shared workspaces - false

Adding an Authorization Manager

If you want to use an authorization service with your repository, you write a custom authorization manager class and add the element tags and values for the custom class within the <RepositoryConfig/> tags. When the repository is created, the element tags for the authorization manager are generated in the `com.blazesoft.repository_config.cfg` file located in the root directory.

For information on writing a custom authorization manager class, see “[Implementing an Authorization Manager](#)” on page 389.

The Authentication and Authorization Example uses a repository configured with a custom authorization manager class. See `<ADVISOR_HOME>/examples/repository/AuthorizationManagerExampleRepository`.

If you are changing your existing repository configuration, you need to run the `NdRomAdminUtil` utility. See “[Changing an Existing Repository Configuration](#)” on page 144.

Note For reasons of brevity, a pair of element tags will be represented by the following notation, <example>.

Tag	Description
<RepositoryAuthorizationManagerFactory/>	Used to create an instance of the class
<JavaName/>	The default class name is: <code>com.blazesoft.repository.base.NdRepositoryDefaultAuthorizationManager</code> See <code>com.blazesoft.repository.base.NdRepositoryAuthorizationManager</code> and its default implementation, <code>NdDefaultRepositoryAuthorizationManager</code> class in the API Reference > Innovator (Frames version or Non-frames version).

Example for adding an authorization service to an existing configuration:

```
<?xml version="1.0" encoding="UTF-8" ?>
<RomConfig>
    <RepositoryConfig>
        <! AtticRepository tags are auto-generated for most repository types-->
        <! and its value of true is only relevant for a BVS shared repository -->
        <AtticRepository> false </AtticRepository>
        <RepositoryAuthorizationManagerFactory>
            <JavaName> examples.CustomAuthorizationManager </JavaName>
        </RepositoryAuthorizationManagerFactory>
        ...
    </RepositoryConfig>
</RomConfig>
```

 **Note** If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Adding an Authentication Service for IP Protection

If you want to use the IP protection framework with your versioned repository, you write a custom authentication service class and an encryption converter class. You manually add the element tags and values to reference the custom classes within the `<RepositoryConfig/>` tags in the `com.blazesoft.repository_config.cfg` file located in the repository or workspace directory. You do not need to run the `NdRomAdminUtil` utility to change the configuration in this case.

For information on writing the custom authentication service classes, see “[Implementing Content Protection for Repository Items](#)” on page 413.

The Authentication and Authorization Example uses a repository configured the IP protection framework. See `<ADVISOR_HOME>/examples/repository/AuthorizationManagerExampleRepository`.

After you write the classes and edit your repository configuration, you need to sign your repository configuration by running the `NdRomAdminUtil` utility. See “[Embedding a Digital Signature Into a Configuration](#)” on page 157.

You can also verify the signature in a repository configuration using the `NdRomAdminUtil` utility. See “[Verifying a Digital Signature](#)” on page 159.

Note For reasons of brevity, a pair of element tags will be represented by the following notation, `<example/>`.

Tag	Description
<code><SecurityConfig/></code>	Used to configure the authentication service for the IP protection framework in your repository or workspace configuration. Add this tag after the <code><RepositoryVersionManagerConfig/></code> end tag.
<code><AuthenticationService Connection/></code>	Used to configure the authentication service connection.
<code><Factory/></code>	Used to create an instance of the authentication service repository connection. The default class is: <code>com.blazesoft.repository.security.NdDefaultAuthenticationServiceConnection</code>
<code><PrivilegedIdentityToken/></code>	Used to specify the token used by privileged users to connect to the repository.
<code><AuthenticationService Factory/></code>	Used to configure the authentication service.
<code><JavaName/></code>	Used to create an instance of the authentication service.
<code><EncryptedContentConverter Factory/></code>	Used to configure the Encryption Converter class.
<code><JavaName/></code>	Used to create an instance of the Encryption converter.

Example for adding an authentication service to an existing configuration:

```

<?xml version="1.0" encoding="UTF-8" ?>
<RomConfig>
...
<RepositoryConfig>
<! AtticRepository tags are auto-generated for most repository types-->
<!and its value of true is only relevant for a BVS shared repository -->

    <AtticRepository> false </AtticRepository>
...
<SecurityConfig>
    <AuthenticationServiceConnection>
        <Factory> com.blazesoft.repository.security.NdDefaultAuthentication
                    ServiceConnection </Factory>
        <PrivilegedIdentityToken> root </PrivilegedIdentityToken>
    </AuthenticationServiceConnection>

    <AuthenticationServiceFactory>
        <JavaName> examples.ExampleAuthenticationService </JavaName>
    </AuthenticationServiceFactory>
    <EncryptedContentConverterFactory>
        <JavaName> examples.ExampleEncryptedContentConverter </JavaName>
    </EncryptedContentConverterFactory>
</SecurityConfig>
...
</RepositoryConfig>
</RomConfig?

```



Note If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Adding a Repository Entry Filter

If you want to remove your repositories entries, you can use a custom repository entry filter class. For example, if you are using an external Source Control Management (SCM), you may want to write a custom class to temporarily remove the version tracking artifacts so that they are not displayed your workspace in the Blaze Advisor IDE. You add the element tags and values for a repository entry filter manager within the `<RepositoryConfig/>` tags. When the repository is created, the element tags for the repository entry filter manager are generated in the `com.blazesoft_repository_config.cfg` file located in the root directory.

If you are changing your existing repository configuration, you need to run the `NdRomAdminUtil` utility. See “[Changing an Existing Repository Configuration](#)” on page 144.

-  **Note** For reasons of brevity, a pair of element tags will be represented by the following notation, `<example>`.

Tag	Description
<code><RepositoryEntryFilterFactory></code>	Used to create an instance of the class <code>NdRepositoryEntryFilter</code> . You add a repository entry filter when you want to filter any administrative files that a source control system has placed in a directory from displaying in the repository.
<code><JavaName></code>	To write a custom repository entry filter factory, see <code>com.blazesoft.repository.base.NdRepositoryEntryFilter</code> in the API Reference > Innovator (Frames version or Non-frames version).

Example for adding an entry filter to an existing configuration:

```
<?xml version="1.0" encoding="UTF-8" ?>
<RepositoryConfig>
    <! AtticRepository tags are auto-generated for most repository types-->
    <!and its value of true is only relevant for a BVS shared repository -->
    <AtticRepository> false </AtticRepository>
    <RepositoryEntryFilterFactory>
        <JavaName>
            examples.CustomEntryFilter
        </JavaName>
    </RepositoryEntryFilterFactory>
    ...
</RepositoryConfig>
```

-  **Note** If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Adding an Item Content Converter

If you want to convert or encrypt your repository contents, you write a custom item content converter class to convert content on read and write operations. You add the custom class using the element tags and values for a version manager within the `<RepositoryConfig>` element tags. When the repository is created, the element tags for the item content converter manager are generated in the `com.blazesoft.repository_config.cfg` file located in the root directory.

If you are changing your existing repository configuration, you need to run the `NdRomAdminUtil` utility. See “[Changing an Existing Repository Configuration](#)” on page 144.

- Note** For reasons of brevity, a pair of element tags will be represented by the following notation, `<example>`.

Tag	Description
<RepositoryItemContentConverterFactory/>	Used to create an instance of the class
<JavaName/>	To write a custom item content converter, see com.blazesoft.template.repository.base.NdRepositoryItemContentConverter in the API Reference > Innovator (Frames version or Non-frames version).

Example for adding an item content converter to an existing configuration:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RomConfig>
...
    <RepositoryConfig>
        <! AtticRepository tags are auto-generated for most repository types-->
        <!and its value of true is only relevant for a BVS shared repository -->
        <AtticRepository> false </AtticRepository>
        <RepositoryItemContentConverterFactory>
            <JavaName>
                examples.CustomItemContentConverter
            </JavaName>
        </RepositoryItemContentConverterFactory>
...
    </RepositoryConfig>
</RomConfig>
```



Note If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Adding a Query

By default the com.blazesoft.template.repository.query.NdRomDefaultQueryManager is query manager is added for you in the com.blazesoft_rom_config.cfg located in the root directory. You can add a custom query to your repository configuration by adding element tags within the <RomConfig/> tags. When the repository is created, the element tags for the additional query are generated in the com.blazesoft_rom_config.cfg file.

If you are changing your existing repository configuration, you need to run the NdRomAdminUtil utility. See “[Changing an Existing Repository Configuration](#)” on page 144.



Note For reasons of brevity, a pair of element tags will be represented by the following notation, <example/>.

Tag	Description
<RomQueryManagerConfig/>	Used to configure the Query Manager.
<RomQueryManagerFactory/>	Used for the Query Manager Factory.

<JavaName/>	The default Java class is: com.blazesoft.template.repository.query.NdRomDefaultQueryManager
<RomQueryFactory/>	(Optional) If you want to add a custom query to the repository configuration. You add the Query Factory tag and specify the package name for the class between the <JavaName/> tags. You can add one or more custom query classes.
<JavaName/>	(Optional) If you want to write a custom query, see com.blazesoft.template.repository.query in the API Reference > Innovator (Frames version or Non-frames version).

Example for adding a query manager to an existing configuration:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RomConfig>
    <RepositoryConfig>
        ...
        </RepositoryConfig>
        <RomQueryManagerConfig>
            <RomQueryManagerFactory>
                <JavaName>
                    com.blazesoft.template.repository.query.NdRomDefaultQueryManager
                </JavaName>
            </RomQueryManagerFactory>
            <RomQueryFactory>
                <JavaName> examples.CustomQuery1</JavaName>
            </RomQueryFactory>
            <RomQueryFactory>
                <JavaName> examples.CustomQuery2</JavaName>
            </RomQueryFactory>
        </RomQueryManagerConfig>
    </RomConfig>
```



Note If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Adding a Filter

By default, Blaze Advisor provides a Standard Filter Instance that you can use to create a filter to use with your projects. See “Filtering Project Content” in *DevelopingRuleProjects.pdf*.

If you want to create your own filter, you write a custom filter class and add the element tags and values for a extractor filter within the <RomConfig/> tags. When the repository is created, the element tags for the filter manager are generated in the `com_blaresoft_rom_config.cfg` file located in the root directory.



Note If you want to add an extractor filter that uses a management properties filter, see `com.blazesoftware.template.repository.deploy.NdPromManagementPropertyFilter` class in the API Reference > Innovator (Frames version or Non-frames version).

If you are changing your existing repository configuration, you need to run the `NdRomAdminUtil` utility. See "["Changing an Existing Repository Configuration" on page 144.](#)



Note For reasons of brevity, a pair of element tags will be represented by the following notation, `<example>`.

Tag	Description
<code><RomExtractorFilterConfig/></code>	Used to configure the Filter Manager.
<code><RomExtractorFilterFactory/></code>	Used to create an instance of the class
<code><JavaName/></code>	To write a custom filter, you need to implement the <code>NdRomExtractorFilter</code> interface and one of the following sub-interfaces in the same package: <code>NdPromItemFilter</code> , <code>NdPromEntityFilter</code> , or <code>NdPromEntityContentFilter</code> , see <code>com.blazesoftware.template.repository.deploy.NdRomExtractorFilter</code> in the API Reference > Innovator (Frames version or Non-frames version).

Example for adding a filter manager to an existing configuration:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RomConfig>
    <RepositoryConfig>
        ...
        </RepositoryConfig>
        <RomExtractorFilterConfig>
            <RomExtractorFilterFactory>
                <JavaName>
                    com.blazesoftware.template.repository.deploy.NdPromEntityTypeFil
                    ter
                </JavaName>
            </RomExtractorFilterFactory>
        </RomExtractorFilterConfig>
    </RomConfig>
```



Note If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example Configuration Files

If you want to write your own configuration files, you can use those files or the following examples as a starting point.

- ["Repository Configuration With No Services" on page 202](#)
- ["Repository Configuration with BVS Versioning" on page 202](#)

- “[Repository Configuration with File CVS Versioning](#)” on page 204
- “[File Repository \(Subversion versioning\)](#)” on page 179
- “[File Repository \(ClearCase versioning\)](#)” on page 181

If you want to add an authorization service to a repository configuration, see [“Implementing an Authorization Manager”](#) on page 389.

Repository Configuration With No Services

The following tags can be used in a configuration files for creating a file or Database repository with no versioning or authorization services:



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RomConfig>
    <AllowUnusedValues> true </AllowUnusedValues>
    <RomConnectionManagerConfig>
        <ConnectionMode> 0 </ConnectionMode>
        <RomConnectionManagerFactory>
            <JavaName>
                com.blazesoft.template.repository.impl.NdDefaultRom
                ConnectionManager</JavaName>
            </RomConnectionManagerFactory>
        </RomConnectionManagerConfig>
    <RomQueryManagerConfig>
        <RomQueryManagerFactory>
            <JavaName>
                com.blazesoft.template.repository.query.NdRomDefault
                QueryManager</JavaName>
            </RomQueryManagerFactory>
        </RomQueryManagerConfig>
    <RomSchemaManagerConfig>
        <RomSchemaManagerFactory>
            <JavaName>
                com.blazesoft.template.repository.impl.NdDefault
                RomSchemaManager
            </JavaName>
        </RomSchemaManagerFactory>
    </RomSchemaManagerConfig>
</RomConfig>
```

Repository Configuration with BVS Versioning

You can use this repository configuration file when creating a file or Database repository with BVS versioning. The example below is for a BVS repository with a private workspace. This example is similar to the romConfig.cfg in the \lib directory of your Blaze Advisor

installation. See “[Sample Connection and Configuration Files Installed With Blaze Advisor](#)” on page 170.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RomConfig>
    <AllowUnusedValues> true </AllowUnusedValues>
    <RomConnectionManagerConfig>
        <ConnectionMode> 0 </ConnectionMode>
        <RomConnectionManagerFactory>
            <JavaName>
                com.blazesoft.template.repository.impl.NdDefaultRomConnectionManager
            </JavaName>
        </RomConnectionManagerFactory>
        </RomConnectionManagerConfig>
    <RomQueryManagerConfig>
        <RomQueryManagerFactory>
            <JavaName>
                com.blazesoft.template.repository.query.NdRomDefaultQueryManager
            </JavaName>
        </RomQueryManagerFactory>
    </RomQueryManagerConfig>
    <RomSchemaManagerConfig>
        <RomSchemaManagerFactory>
            <JavaName>
                com.blazesoft.template.repository.impl.NdDefaultRomSchemaManager
            </JavaName>
        </RomSchemaManagerFactory>
    </RomSchemaManagerConfig>
    <RepositoryConfig>
        <! AtticRepository tags are auto-generated for most repository types-->
        <!and its value of true is only relevant for a BVS shared repository -->
        <AtticRepository> false </AtticRepository>
        <RepositoryVersionManagerConfig>
            <RepositoryVersionManagerFactory>
                <JavaName>
                    com.blazesoft.repository.generic.version.NdNativeRepositoryVersion
                        Manager
                </JavaName>
            </RepositoryVersionManagerFactory>
            <Impersonate> true </Impersonate>
            <PrivateWorkspaceSupported> true </PrivateWorkspaceSupported>
        </RepositoryVersionManagerConfig>
    </RepositoryConfig>
    ...
</RomConfig>
```

-  **Note** If you want to create a BVS repository with a shared workspace, you would change the value between the <AtticRepository/> tags from false to true and change the value between the <PrivateWorkspaceSupported/> tags from true to false.

Repository Configuration with File CVS Versioning

You can use this repository configuration file when creating a File repository with CVS versioning.

-  **Note** To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<RomConfig>
  <AllowUnusedValues> true </AllowUnusedValues>
  <RomConnectionManagerConfig>
    <ConnectionMode> 0 </ConnectionMode>
    <RomConnectionManagerFactory>
      <JavaName>
        com.blazesoftware.template.repository.impl.NdDefaultRomConnectionManager
      </JavaName>
    </RomConnectionManagerFactory>
  </RomConnectionManagerConfig>
  <RomQueryManagerConfig>
    <RomQueryManagerFactory>
      <JavaName>
        com.blazesoftware.template.repository.query.NdRomDefaultQueryManager
      </JavaName>
    </RomQueryManagerFactory>
  </RomQueryManagerConfig>
  <RomSchemaManagerConfig>
    <RomSchemaManagerFactory>
      <JavaName>
        com.blazesoftware.template.repository.impl.NdDefaultRomSchemaManager
      </JavaName>
    </RomSchemaManagerFactory>
  </RomSchemaManagerConfig>
  <RepositoryConfig>
    <! AtticRepository tags are auto-generated for most repository types-->
    <! and its value of true is only relevant for a BVS shared repository -->
    <AtticRepository> true </AtticRepository>
    <RepositoryVersionManagerConfig>
      <RepositoryVersionManagerFactory>
        <JavaName>
          com.blazesoftware.repository.file.NdFileCVSWorkspaceVersionManager
        </JavaName>
      </RepositoryVersionManagerFactory>
      <RepositoryVersionParserFactory>
        <JavaName>
          com.blazesoftware.repository.file.NdDefaultCVSRepositoryVersionResult
          Parser
        </JavaName>
      </RepositoryVersionParserFactory>
    </RepositoryVersionManagerConfig>
  </RepositoryConfig>
</RomConfig>
```

```

</JavaName>
</RepositoryVersionParserFactory>
<PrivateWorkspaceSupported> true </PrivateWorkspaceSupported>
</RepositoryVersionManagerConfig>
</RepositoryConfig>
...
</RomConfig>
```

File Repository (Subversion Versioning)

You can use this repository configuration file when creating a File repository (Subversion versioning).



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```

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

<RomConfig>
    <RomQueryManagerConfig>
        <RomQueryManagerFactory>
            <JavaName> com.blazesoft.template.repository.query.
                NdRomDefaultQueryManager </JavaName>
            </RomQueryManagerFactory>
        </RomQueryManagerConfig>
        <RomConnectionManagerConfig>
            <ConnectionMode> 0 </ConnectionMode>
            <RomConnectionManagerFactory>
                <JavaName> com.blazesoft.template.repository.impl.NdDefault
                    RomConnectionManager</JavaName>
                </RomConnectionManagerFactory>
            </RomConnectionManagerConfig>
            <RomSchemaManagerConfig>
                <RomSchemaManagerFactory>
                    <JavaName> com.blazesoft.template.repository.impl.NdDefault
                        RomSchemaManager </JavaName>
                    </RomSchemaManagerFactory>
                </RomSchemaManagerConfig>
                <RepositoryConfig>
                    <RepositoryVersionManagerConfig>
                        <RepositoryVersionManagerFactory>
                            <JavaName>com.blazesoft.repository.scm.subversion.NdSvnWorkspace
                                VersionManager</JavaName>
                        </RepositoryVersionManagerFactory>
                        <PrivateWorkspaceSupported> true </PrivateWorkspaceSupported>
                        <VersioningRepositoryConnection>
                            <Factory> com.blazesoft.repository.scm.NdScmServerConnection
                                </Factory>
                            <RepositoryVersionSystemAdminFactory>
                                <JavaName>com.blazesoft.repository.scm.subversion.NdSvnVersionSystemAdmin
                                    </JavaName>
                                </RepositoryVersionSystemAdminFactory>
                            </RepositoryVersionSystemAdminFactory>
                        </VersioningRepositoryConnection>
                    </RepositoryVersionManagerConfig>
                </RepositoryConfig>
            </RomSchemaManagerFactory>
        </RomConnectionManagerConfig>
    </RomQueryManagerConfig>
</RomConfig>
```

```
<VersionManagerClass>com.blazesoft.repository.scm.NdFileScmVersioning
    SystemVersionManager</VersionManagerClass>
    <User> user1 </User>
    <Password> pswd </Password>
    <RepositoryName> repoTest1 </RepositoryName>
    <ServerUrl> svn://<serverhostname>/<path to the parent directory
        either branch or trunk>
    </VersioningRepositoryConnection>
    <Name> Version Control Configuration for CC </Name>
    </RepositoryVersionManagerConfig>
    <! AtticRepository tags are auto-generated for most repository types-->
    <!and its value of true is only relevant for a BVS shared repository -->
    <AtticRepository> false </AtticRepository>
</RepositoryConfig>

    <AllowUnusedValues> true </AllowUnusedValues>

</RomConfig>
```

File Repository (ClearCaseVersioning)

You can use this repository configuration file when creating a File repository (ClearCase versioning) connected to an IBM Rational ClearCase server.

-  **Note** To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RomConfig>
    <RomQueryManagerConfig>
        <RomQueryManagerFactory>
            <JavaName> com.blazesoft.template.repository.query.
                NdRomDefaultQueryManager </JavaName>
            </RomQueryManagerFactory>
    </RomQueryManagerConfig>
    <RomConnectionManagerConfig>
        <ConnectionMode> 0 </ConnectionMode>
        <RomConnectionManagerFactory>
            <JavaName> com.blazesoft.template.repository.impl.NdDefault
                RomConnectionManager</JavaName>
            </RomConnectionManagerFactory>
    </RomConnectionManagerConfig>
    <RomSchemaManagerConfig>
        <RomSchemaManagerFactory>
            <JavaName> com.blazesoft.template.repository.impl.NdDefault
                RomSchemaManager </JavaName>
            </RomSchemaManagerFactory>
    </RomSchemaManagerConfig>
    <RepositoryConfig>
        <RepositoryVersionManagerConfig>
            <RepositoryVersionManagerFactory>
```

```
<JavaName> com.blazesoft.repository.scm.wvcm.cc.NdCcVersion
Manager </JavaName>
</RepositoryVersionManagerFactory>
<PrivateWorkspaceSupported> true </PrivateWorkspaceSupported>
<VersioningRepositoryConnection>
<Factory> com.blazesoft.repository.scm.NdScmServerConnection </
Factory>
<RepositoryVersionSystemAdminFactory>
<JavaName>com.blazesoft.repository.scm.wvcm.NdFileWvcmVersion
SystemAdmin</JavaName>
</RepositoryVersionSystemAdminFactory>
<VersioningSystemVersionManagerFactory>
<JavaName>com.blazesoft.repository.scm.wvcm.cc.NdCcVersioning
SystemVersionManager</JavaName>
</VersioningSystemVersionManagerFactory>

<VersionManagerClass>com.blazesoft.repository.scm.wvcm.cc.NdCcVersion
Manager</VersionManagerClass>

<ScmProvider>com.ibm.rational.stp.client.internal.cc.CcOnlyProviderImpl
</ScmProvider>
<WorkspaceFolder>C:\ccview</WorkspaceFolder>
<RepositoryHomeFolder>/vobs/myvob</RepositoryHomeFolder>
<User> user1 </User>
<Password> pswd </Password>
<RepositoryName> repoTest1 </RepositoryName>
<ServerUrl> http://serverHost/ccrc </ServerUrl>
</VersioningRepositoryConnection>
<Name> Version Control Configuration for CC </Name>
</RepositoryVersionManagerConfig>
<! AtticRepository tags are auto-generated for most repository types-->
<!and its value of true is only relevant for a BVS shared repository -->
<AtticRepository> false </AtticRepository>
</RepositoryConfig>

<AllowUnusedValues> true </AllowUnusedValues>

</RomConfig>
```


CHAPTER 6

Implementing a Repository With a Blaze Versioning Service

In the FICO® Blaze Advisor® decision rules management system if you want to use a versioning service with a repository but you do not want to use an external Source Control Management (SCM) system, you can create a repository with a Blaze Versioning Service (BVS). Your users can connect to a repository where they can use common versioning commands to check out, check in, update, or check the status of files. Each time users check files back into the repository, the revision history is updated to record the action. You can create a File or Database repository with a BVS versioning service. This topic includes the following sections:

- ["Deciding on the Type of BVS Workspace" on page 209](#)
- ["Creating a BVS Repository Using the Blaze Advisor IDE" on page 216](#)
- ["Creating a Local or Private Workspace" on page 226](#)
- ["Verifying Blaze Advisor Versioning Commands" on page 230](#)

If you want to use BVS versioning with a Relational Database Management System (RDBMS) or a NoSQL database, see:

- Database (RDBMS) Repositories
["Implementing a Database Repository" on page 285](#)
- MongoDB (NoSQL) Repositories
["Implementing MongoDB Repositories" on page 309](#)

If you decide that you want to use an external SCM system, Blaze Advisor supports File CVS, ClearCase, or Subversion repositories.

- ["Implementing a CVS Repository" on page 235](#)
- ["Implementing External SCM Repositories" on page 253](#)

If you want to add an authorization service, see ["Implementing an Authorization Manager" on page 389](#).

Deciding on the Type of BVS Workspace

A workspace is a directory on your local machine where you do most of the work related to developing and testing a rule service. If you want to use a Blaze Versioning system, you can choose a File, Database (RDBMS), or MongoDB(NoSQL) repository type.

If you choose a File repository type, you have a choice between two workspace types: *shared* or *private*.

If you choose a Database repository type, you use a private workspace.

If you choose a MongoDB repository type, you have several choices of workspace types: *shared* MongoDB workspace, *private* MongoDB workspace, or *private* File workspace.

- If you implement a File, Database (RDBMS), or MongoDB BVS repository with a private File workspace, your users will connect to a repository located on a shared network location and will work on files in isolation in a local directory. A private File workspace is equivalent to a local workspace in many external SCM systems. See ["Private Workspaces" on page 211](#).

Users can control when they are exposed to changes to files made by others. They will work on their files without being aware of any changes until they update their workspace. They can check in their files when they are ready for others to have access to their changes.

When users connect to a BVS repository with private File workspaces, the connection allows them access to the repository and a workspace. The default type for a private workspace is file. However, it is possible to create workspaces that use one of the other types supported by Blaze Advisor when connected to the Blaze Advisor IDE. See ["Writing a Connection Configuration File for a Private or a Local Workspace" on page 185](#).

- If you implement a MongoDB BVS repository with private MongoDB workspaces, your users will connect to a repository in MongoDB and a workspace also in MongoDB. The default type for a private workspace is file. However, it is possible to create workspaces that use one of the other types supported by Blaze Advisor when connected to the Blaze Advisor IDE. See ["Writing a Connection Configuration File for a Private or a Local Workspace" on page 185](#).
- If you implement a File BVS repository with a shared workspace, your users will connect to a repository and a workspace in the same physical location. Your users can view the changes made by others as soon as the files are saved or checked in by refreshing their repository or by updating their workspace. See ["Shared Workspaces" on page 213](#).
- If you implement a MongoDB BVS repository with a shared workspace, your users will connect to a repository and a workspace in MongoDB. Your users can view the changes made by others as soon as the files are saved or checked in by refreshing their repository or by updating their workspace. See ["Shared Workspaces" on page 213](#).

Users of a shared, private File workspaces, or private MongoDB workspaces can obtain the most current, read-only versions of project files by refreshing or updating their workspace. If they want to make changes to files, they need to check them out, make their changes, and then check them back into the repository. The version history of every project or project item file is stored in a `<ItemName>_version_info.xml`. The information in

this file can be retrieved and displayed in the History tab of an editor in the Blaze Advisor IDE or an RMA.

Private Workspaces

Like local workspaces that are used with the supported external SCM types like the CVS, Subversion or ClearCase repository, private file or MongoDB workspaces provide user-controlled isolation from file changes made by others. These types of workspaces can be advantageous when it is important to control how and when changes made to files are available to others and to control when files changed by others are made available to you. Each private workspace is associated with a specific user identifier and no other users can gain access to the repository contents. For information about local workspaces, see ["Local Workspaces" on page 238](#).

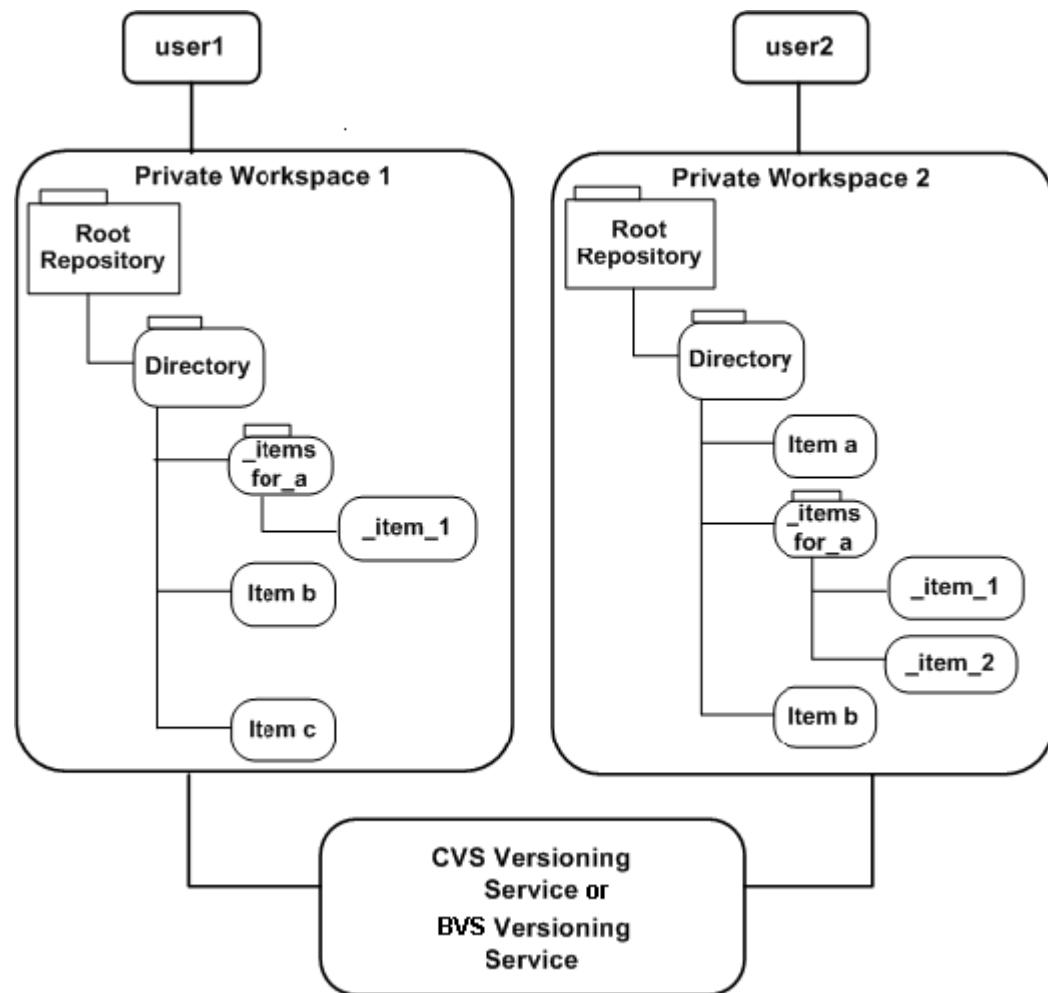
By default, with the exception of the MongoDB repository with private MongoDB workspaces, most private workspaces are of file type. However, you can create a workspace that is one of the other types supported by Blaze Advisor. See ["Writing a Connection Configuration File for a Private or a Local Workspace" on page 185](#).

You can need to explicitly create a new repository connection in Blaze Advisor. At any time, users can obtain the latest versions of files by updating their workspace. Users interact with the BVS versioning service in order to check out files, check in any changes to files, and to obtain updates, version history, and item status. The contents and structure of each private workspace can differ.

File Structure for a Private or Local Workspace

Each time users connect to a BVS repository (File, Database (RDBMS), or MongoDB(NoSQL)) with a private file workspace , a CVS or an Subversion or ClearCase repository with a local workspace in Blaze Advisor, they can view the contents of their private or local workspace in the Repository Explorer. In this diagram you see two users connected to the same repository with BVS versioning service. Each user is also connected to a private or local workspace.

-  **Note** The contents of each user's workspace is not identical. user1 has created *Item c* in Workspace 1, this file is not available to user2 until user1 checks it into the BVS, CVS, Subversion, or ClearCase versioning service. Likewise, user2 has *Item a* which not available to user1 until user2 checks it in to the repository.



This table describes the contents of each private workspace and their relationship to the versioning service.

Diagram Key	Description
user n	Multiple users can connect to the same BVS, CVS, or ClearCase or Subversion repository. Each user edits files that have been checked out to a private or local workspace located in a local directory.
Workspace n	<p>Users connected to the same repository work independently in private or local workspaces.</p> <p>A private or local workspace allows a user to update their workspace, check out the file or files they want to work on, test the changes locally and when they are satisfied with the changes, check them into the repository. During the time that a user has a file or files checked out of the repository, it cannot be checked out by another user. This prevents two users from making changes to the same file and overwriting the changes made by the other user.</p> <p>Other users connected to the same repository can update their private or local workspace with the most current version of the files that have been checked into the repository. If a file or files have been checked out by another user, users will see a read-only version of the file or files.</p>
Workspace Directory	The path name to the directory where the private or local workspace is located.
Project Folder	The project folder is used to organize project item files. A directory folder contains its own files that are used to contain information about the file and its track revision history.
proj_item n	The proj_item file represents project items that are added to a project and checked into the versioning service.
_Items for_x	This folder is created if the user chooses to manage the rule instance files separately from the ruleset instance file. The _Items for_x folder contains its own BVS files to manage the version history of rule instances.
_subitem_x	This is an individual rule instance. Each rule instance is assigned a unique identifier when you specify that rule instances be stored separately.
BVS versioning service, CVS, ClearCase, or Subversion versioning service	The BVS, CVS or ClearCase or Subversion versioning service contains the latest version of all files that have been checked in. When users connect to their respective private or local workspaces, they connect to their repository.

Shared Workspaces

You can configure a File BVS repository or a MongoDB BVS repository with a shared workspace when you want your users to be aware of the changes made by others connected to the same repository and workspace. Users are aware of changes made by

others because the workspace is not a separate directory location from the repository. See ["The Blaze Advisor Repositories and Workspaces" on page 21](#).

Using a shared workspace can be advantageous when it is important to know when and what changes are being made by other users. For example, if two users are connected to a BVS repository with a shared workspace and user 1 has checked out a file, user 2 can refresh their repository and see that the file has been checked out. When user1 saves a change and user2 refreshes their repository, they can see a read-only version of the change made by user1 even though the file is still locked. Users who are connected to a shared workspace do not work in complete isolation and the status of the files including the changes that have been saved but not checked can be known by the rest of the development team.

In a File BVS repository with a shared workspace, an Attic folder is added to each directory and subfolder. Each Attic folder contains files with information about the project item including its version history. See ["Creating a File Repository With a BVS Shared Workspace Using the Blaze Advisor IDE" on page 219](#).

In a MongoDB BVS repository with a shared workspace, the revision tracking is handled with the MongoDB instance. See ["Creating a MongoDB Repository \(BVS versioning - shared\) using the Blaze Advisor IDE" on page 334](#).

File Structure for a BVS Shared Workspace

Each time users connect to a File BVS repository with a shared workspace in Blaze Advisor, they can view the contents of shared workspace in the Repository Explorer. If the contents are currently checked out by another user, the contents are read-only. In this diagram you see three users connected to the same File BVS repository with a shared workspace.



Note Each user's view of the workspace is identical assuming that they regularly refresh their repository contents.

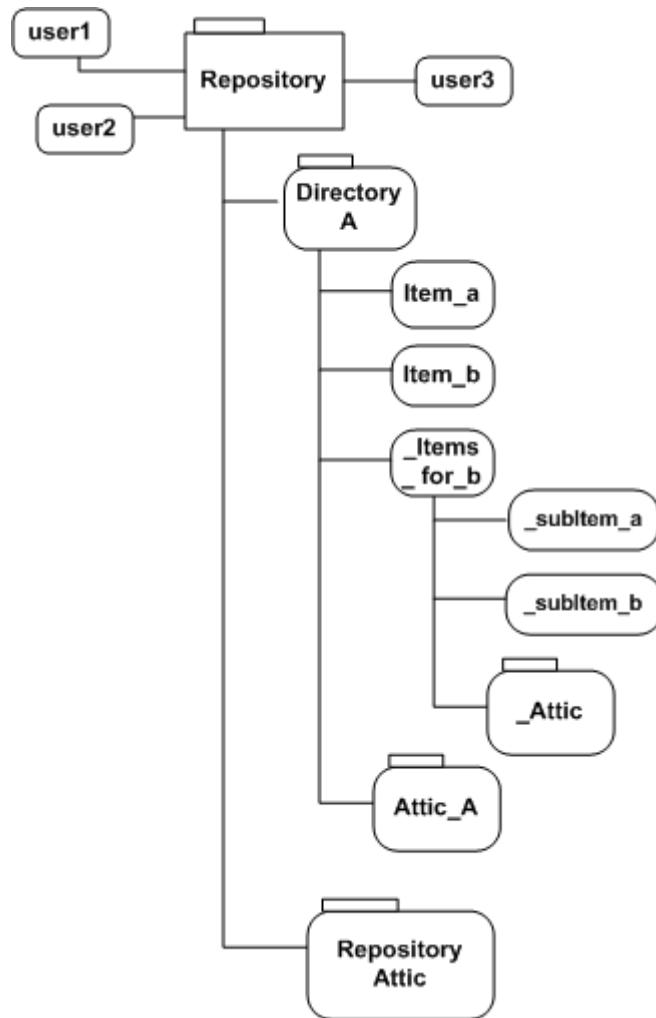


Diagram Key	Description
user1, user2, user3	<ul style="list-style-type: none"> Multiple users can connect to the same repository. In a repository with File BVS shared workspace, each user who connects to the same repository. This means that each user can view the status of all files as well as any changes made by other users as soon as they are saved, checked out, or checked in.
Repository	The repository is configured to use File Blaze Versioning with a shared workspace.
DirectoryA	<p>The DirectoryA folder and the Repository Attic folder appear at the top level in the repository. The Repository Attic manages the version history of top-level files in the Repository.</p> <p>DirectoryA contains three project items and an Attic folder. Attic_A manages the version history of the project items in the DirectoryA folder.</p>

Diagram Key	Description
Repository Attic, Attic_A, _Attic	<p>The Attic folders contain files that are used to manage item revisions. Depending on the structure of the folders and files in your repository, you can have several directory folders or subfolders that each have their own Attic folder.</p> <ul style="list-style-type: none">■ Each time you check in a file revision this change is recorded in the corresponding Attic folder.■ When you view the version history of a file, the information from the Attic folder for that file is retrieved.■ When you create a new repository with File Blaze Versioning with a shared workspace, or convert an existing repository to use File Blaze Versioning with a shared workspace, an Attic folder is generated.
Item_a, Item_b	Project items can be global-level SRL entities, global-level templates, instances, and global-level providers.
_Items_for_b	The _Items_for_b folder contains rule instances and an _Attic folder to manage rule instance revisions.
subitem_a, subitem_b	These are individual rule instances. Each rule instance is assigned a unique identifier when you specify that rule instances be stored separately.

Creating a BVS Repository Using the Blaze Advisor IDE

You can create a File BVS repository with private workspace or a shared workspace using the Blaze Advisor IDE:

- ["Creating a File Repository With BVS Private Workspaces Using the Blaze Advisor IDE" on page 217](#)
- ["Creating a File Repository With a BVS Shared Workspace Using the Blaze Advisor IDE" on page 219](#)

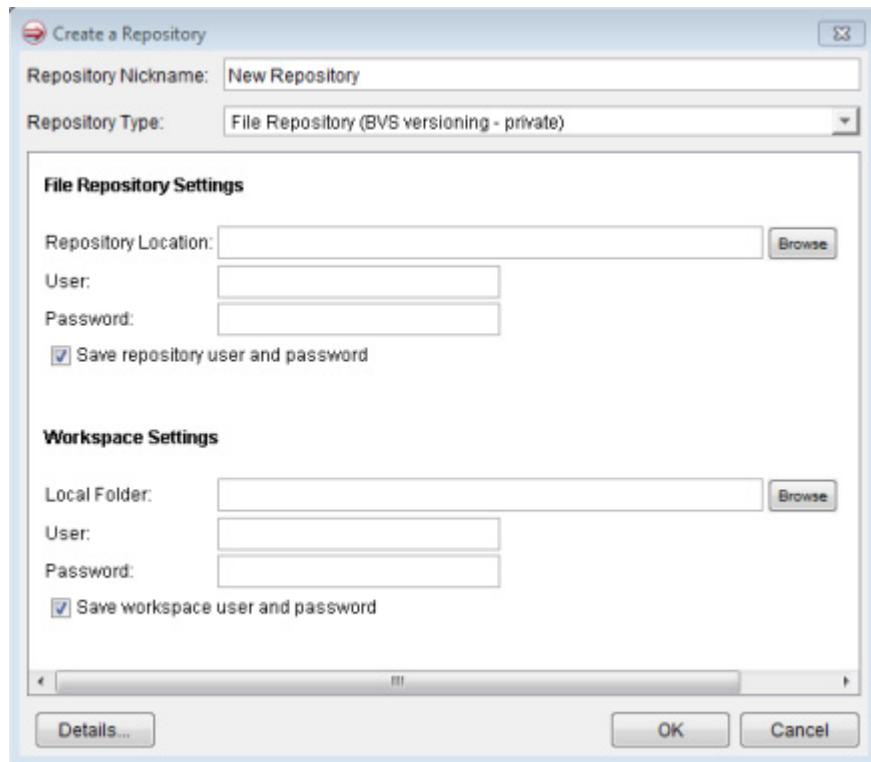
To create a File BVS repository using the utility, see ["Creating a File BVS Repository Using the Utility" on page 220](#).

Creating a File Repository With BVS Private Workspaces Using the Blaze Advisor IDE

If you want to create a File BVS repository with private workspaces, you can create the repository, workspace, and connection at the same time using the Blaze Advisor IDE. If you need to create any additional workspaces, see "["Creating a Local or Private Workspace" on page 226](#).



Note If you plan to create a BVS repository using private workspaces on a remote server, for best performance results, we recommend that you uncheck the Check versioned repository status on the General page of the **Window > Preferences > Blaze Advisor**. When you uncheck this option, only the local workspace status is checked avoiding the status checks to the repository via the network. See "Blaze Advisor General Preferences" in *DevelopingRuleProjects.pdf*.



To create a File BVS repository with a private workspace

- 1 Select **Repository > New Repository**.
- 2 Enter a connection name in the **Repository Nickname** field.
This name is displayed in the Repository Explorer and is also referenced when you open a project for editing in the Project Explorer.
- 3 Select a File Repository (BVS versioning - private) for the **Repository Type**
- 4 Enter the following information to connect to the machine that is hosting the server:

Repository Type Connection Parameter	Description
Repository Location	The path where the repository directory is located.
User name	User identifier
Password	(Optional) Specify a password, if you have implemented authentication and authorization controls.
Save repository user and password	(Optional) Select this option to save the user name and password in the connection instance file and the repository configuration. See “Saving User Name and Passwords” on page 43 .
Local Folder	You enter the path to your private workspace directory. This field is displayed when you choose a repository type with a local or private workspace such as a File CVS repository or a BVS repository with a private workspace.
User name	User identifier
Password	(Optional) Specify a password, if you have implemented authentication and authorization controls.
Save workspace user and password	(Optional) Select this option to save the user name and password in the connection instance file and the repository configuration. See “Saving User Name and Passwords” on page 43 .



Important To avoid any creation errors, when entering values into the fields of the Repository Connection Parameters page, leave no trailing spaces.

After you have created the repository with a private workspace, you may want to verify the versioning commands, see [“Verifying Blaze Advisor Versioning Commands” on page 230](#).

When you create a BVS repository with a private workspace, you need to explicitly connect using the Blaze Advisor IDE in order to be working with it. See [“Connecting to a Blaze Advisor Workspace or Repository” on page 52](#).

After you create the repository, proceed to the following tasks:

- If you need to create a new connection, see [“Creating a New Blaze Advisor Workspace or Repository Connection” on page 58](#).
- After you connect to the workspace, you may need to explicitly update your workspace to see the latest versions of all files.

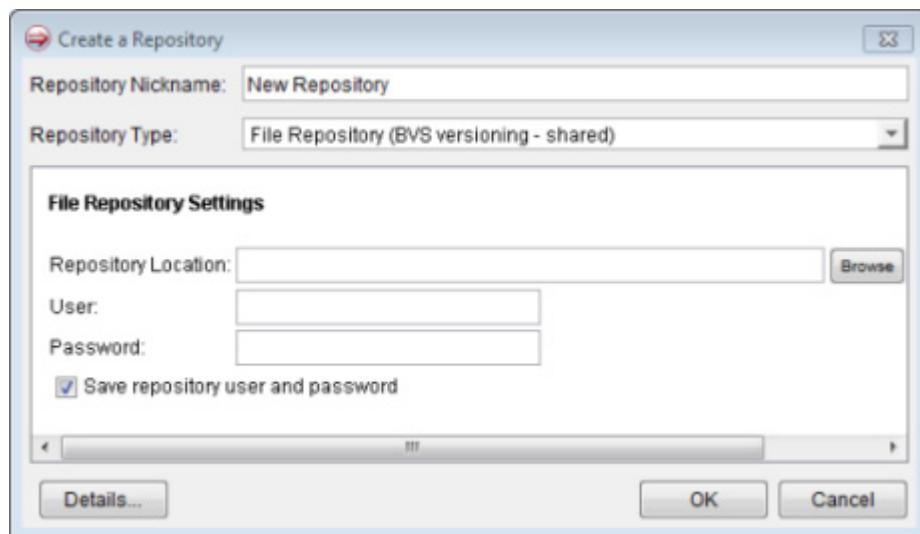
After you create the repository, proceed with one or more of the following tasks:

- If you need to create additional workspaces for your users, see [“Creating a Local or Private Workspace” on page 226](#).

- If you want to create a new workspace connection, see “[Creating a New Blaze Advisor Workspace or Repository Connection](#)” on page 58.
- After you have created the repository with a local workspace, you may want to verify the versioning commands, see “[Verifying Blaze Advisor Versioning Commands](#)” on page 230.

Creating a File Repository With a BVS Shared Workspace Using the Blaze Advisor IDE

If you want to create a File BVS repository with a shared workspace, you can create the repository, workspace, and connection at the same time. The location for the shared workspace is identical to the location of the repository and users all connect to the same location to work on the projects files. Therefore, you do not need to create any additional workspaces for users. However, you can create new connections to the workspace. See “[Creating a New Blaze Advisor Workspace or Repository Connection](#)” on page 58.



To create a BVS repository with a shared workspace

- 1 Select **Repository > New Repository**.
- 2 Enter a connection name in the **Repository Nickname** field.
This name is displayed in the Repository Explorer and is also referenced when you open a project for editing in the Project Explorer.
- 3 Select a File Repository (BVS versioning - shared) for the **Repository Type**
- 4 Enter the following information to connect to the machine that is hosting the server:

Repository Type Connection Parameter	Description
Repository Location	The path where the repository directory is located. This location also represents the location of the shared workspace.
User name	User identifier
Password	(Optional) Specify a password, if you have implemented authentication and authorization controls.
Save repository user and password	(Optional) Select this option to save the user name and password in the connection instance file and the repository configuration. See "Saving User Name and Passwords" on page 43 .



Important To avoid any creation errors, when entering values into the fields of the Repository Connection Parameters page, leave no trailing spaces.

When you create a BVS repository with a shared workspace, you need to explicitly connect using the Blaze Advisor IDE in order to be working with it. See ["Connecting to a Blaze Advisor Workspace or Repository" on page 52](#).

After you create the repository, proceed to the following tasks:

- Use the Repository Explorer to begin creating the overall repository structure with folders and projects.
- If you need to create additional workspaces for your users, see ["Creating a Local or Private Workspace" on page 226](#).
- If you need to create a new connection, see ["Creating a New Blaze Advisor Workspace or Repository Connection" on page 58](#).

After you have created the repository with a shared workspace, you may want to verify the versioning commands, see ["Verifying Blaze Advisor Versioning Commands" on page 230](#).

Creating a File BVS Repository Using the Utility

You can create a File BVS Repository with private workspaces or a File BVS Repository with a shared workspace using the NdRomAdminUtil utility. You need to write connection configuration files and a repository configuration file before running the utility in a command prompt and entering the command and related arguments or using a batch file. You create a File BVS Repository with private workspaces or a shared workspace by entering specific values for supporting private workspaces and storage of revision information in the repository configuration file.

- ["Creating a File Repository With BVS Private Workspaces Using the Blaze Advisor IDE" on page 217](#)

- “[Creating a File Repository With a BVS Shared Workspace Using the Blaze Advisor IDE](#)” on page 219

Creating a File BVS Repository With Private Workspaces Using the Utility

You can create a File BVS repository with private workspaces using the `NdRomAdminUtil` utility. Before you can run the utility to create the repository, you need to write one or more connection configuration files and a configuration file:

- “[About Writing Files for Creating a Blaze Advisor Repository or Workspace Using the Utility](#)” on page 168
- “[Writing the Connection Configuration File for an Admin Repository](#)” on page 33
- “[Writing a Connection Configuration File for a File BVS Repository](#)” on page 221
- “[Writing a Repository Configuration File for a File BVS Repository](#)” on page 223
- “[Writing a Batch File to Create a File BVS Repository](#)” on page 224

After you create the repository, you can use some of these files to perform other tasks using the `NdRomAdminUtil` utility, see “[The NdRomAdminUtil Utility Commands](#)” on page 162.

You create the private workspaces by using the Blaze Advisor IDE or by writing an additional connection configuration file for the workspace and running the utility again, see “[Creating a Local or Private Workspace](#)” on page 226.

After you have created the File BVS repository with private workspaces, you may want to verify the versioning commands, see “[Verifying Blaze Advisor Versioning Commands](#)” on page 230.

Writing a Connection Configuration File for a File BVS Repository

You can write a connection configuration file for a File repository with BVS versioning. This connection file needs to contain the structure and the values described in the table below.



Note For reasons of brevity, a pair of element tags will be represented by the following notation, <example>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<VersioningRepository Connection/>	The outer <code>RepositoryConnection</code> tag is used when you need to create a connection for a repository with versioning. All other tags in this table are nested within this tag.

Tag	Description
<RepositoryConnection/>	The inner RepositoryConnection tag is nested within the outer VersioningRepositoryConnection tag and is used to nest the factory class for the versioning connection.
<Factory/>	The Factory tag is used when you want to create an instance of a connection for BVS versioning, you use the fully qualified class name. For example, com.blazesoft.repository.generic.version.NdNative VersioningRepositoryConnection.
<PersistCredentials/>	<p>(Optional) Used to save the username and password to the connection instance file in the Admin Repository.</p> <p>If you are using a File or Database BVS repository with private workspaces, the value is applied for both the repository connection and the workspace connection.</p> <p>Possible values:</p> <ul style="list-style-type: none">■ false -- do not store connection credentials (default) If you choose not to store the connection information, each time you connect to the repository in the Blaze Advisor IDE, you need to manually enter your user name and password.■ true -- store the connection credentials See “Saving User Names and Passwords Using the Utility” on page 183.
<RepositoryConnection/>	A inner RepositoryConnection tag is added within the outer RepositoryConnection tags to nest the connection parameter tags for the specific repository type using versioning. The tags and the values you enter for the connection parameters are the same as those you would use when writing a connection file for a repository of that type without versioning: <ul style="list-style-type: none">■ “File Repository” on page 172■ “Database Repository” on page 173
<User/>	This User tag is used to specify the user name for the connection to the versioned repository and appears after the closing inner RepositoryConnection tag and before the outer RepositoryConnection end tag. You need to use this tag if you have a repository with BVS versioning. If you are using BVS versioning you also use this tag before the VersioningRepositoryConnection end tag.

Tag	Description
<Password/>	This Password tag is used to specify the password for the connection to the versioned repository and appears after the closing RepositoryConnection tag and before the VersioningRepositoryConnection end tag. You need to use this tag if you have a connection to a Database server or use an Authorization service that requires a password. If you are using a password with your connection, you also use this tag before the VersioningRepositoryConnection end tag.
<RepositoryName/>	This Repository Name tag is used to specify the Repository Connection Name. You need to use this tag if you have a File or Database repository with BVS versioning. If you are using BVS versioning you also use this tag before the VersioningRepositoryConnection end tag.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example for a File repository with BVS versioning:

```
<?xml version="1.0" encoding="UTF-8" ?>
<VersioningRepositoryConnection>
    <Factory>      com.blazesoft.repository.generic.version.NdNative
    VersioningRepositoryConnection</Factory>
    <RepositoryConnection>
        <Factory>
            com.blazesoft.repository.file.NdFileRepositoryConnection
        </Factory>
        <RepositoryFolder>
            C:\Repository\ProjectRepos\TestThisRepo3563
        </RepositoryFolder>
        <RepositoryName>TestThisRepo3563 </RepositoryName>
    </RepositoryConnection>
    <PersistCredentials> true </PersistCredentials>
</VersioningRepositoryConnection>
```

Writing a Repository Configuration File for a File BVS Repository

You can use this repository configuration file when creating a file or Database repository with BVS versioning. The example below is for a BVS repository with a private workspace. This example is similar to the romConfig.cfg in the \lib directory of your Blaze Advisor installation. See ["Sample Connection and Configuration Files Installed With Blaze Advisor"](#) on page 170.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RomConfig>
    <AllowUnusedValues> true </AllowUnusedValues>
    <RomConnectionManagerConfig>
        <ConnectionMode> 0 </ConnectionMode>
        <RomConnectionManagerFactory>
            <JavaName>
                com.blazesoft.template.repository.impl.NdDefaultRomConnectionManager
            </JavaName>
        </RomConnectionManagerFactory>
    </RomConnectionManagerConfig>
    <RomQueryManagerConfig>
        <RomQueryManagerFactory>
            <JavaName>
                com.blazesoft.template.repository.query.NdRomDefaultQueryManager
            </JavaName>
        </RomQueryManagerFactory>
    </RomQueryManagerConfig>
    <RomSchemaManagerConfig>
        <RomSchemaManagerFactory>
            <JavaName>
                com.blazesoft.template.repository.impl.NdDefaultRomSchemaManager
            </JavaName>
        </RomSchemaManagerFactory>
    </RomSchemaManagerConfig>
    <RepositoryConfig>
        <AtticRepository> false </AtticRepository>
        <RepositoryVersionManagerConfig>
            <RepositoryVersionManagerFactory>
                <JavaName>
                    com.blazesoft.repository.generic.version.NdNativeRepositoryVersion
                        Manager
                </JavaName>
            </RepositoryVersionManagerFactory>
            <Impersonate> true </Impersonate>
            <PrivateWorkspaceSupported> true </PrivateWorkspaceSupported>
        </RepositoryVersionManagerConfig>
    </RepositoryConfig>
    ...
</RomConfig>
```



Note If you want to create a BVS repository with a shared workspace, you would change the value between the `<AtticRepository/>` tags from `false` to `true` and change the value between the `<PrivateWorkspaceSupported/>` tags from `true` to `false`.

Writing a Batch File to Create a File BVS Repository

You run the `NdRomAdminUtil` utility twice to create a File BVS repository with a private workspace. First you run the utility to create the repository and then you run it a second time to create the workspace. Each time you run the utility, you must use the correct arguments and options for each task. See “[Creating a Blaze Advisor Workspace Using the Utility](#)” on page 228.

To create a BVS repository with a shared workspace you run the utility once because the repository and the workspace share the same physical location.

Command, Argument, or Option	Description
createRepository	Command to create a repository
-repositoryConnection	Argument takes the path to the location of the repository connection configuration file (.cfg). See "Repository Connection Configuration File Examples" on page 171 . This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
Note Use this argument to create a workspace for a BVS Repository with private workspaces, File CVS Repository File Repository (Subversion versioning) or a File Repository (ClearCase versioning) using an IBM Rational ClearCase server connection.	Argument takes the path to the location of the workspace connection configuration file (.cfg) See "Writing a Connection Configuration File for a Private or a Local Workspace" on page 185 . This file contains information such as the repository name, the workspace connection name, the user name and password for the connection, and the path to the workspace.
-systemconnection	Argument takes the path to the location of the system Folder connection configuration file (.cfg). See "Repository Connection Configuration File Examples" on page 171 . This file contains information such as the name of the repository which in this case is Admin Repository, the location of the Admin Repository and the user name.
--configuration	Argument takes the path to the location of the configuration file(.cfg).See "Writing a Repository Configuration File" on page 191 .
-verbose	Option prints out all details messages into the command console.
-m	Option to add a comment when creating a repository. Note Use double quotes around your comment text.
-debug	Option prints out tracing information into the command console.

Example of a batch file to create a File BVS repository using pathnames:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil
createRepository
-repositoryConnection C:/Repo_Connections/ConnectionFiles/DefaultRepo.cfg
-systemConnection C:/Repo_Connections/ConnectionFiles/sysAdmin.cfg
-configuration C:/Blaze/Advisor/lib/FileBVSRepository.cfg -verbose
```

Creating a File BVS Repository With a Shared Workspace Using the Utility

You can create a File BVS repository with a shared workspace using the `NdRomAdminUtil` utility. Before you can run the utility to create the repository, you need to write one or more connection configuration files and a configuration file:

- “[About Writing Files for Creating a Blaze Advisor Repository or Workspace Using the Utility](#)” on page 168
- “[Writing the Connection Configuration File for an Admin Repository](#)” on page 33
- “[Writing a Connection Configuration File for a File BVS Repository](#)” on page 221
- “[Writing a Repository Configuration File for a File BVS Repository](#)” on page 223
- “[Writing a Batch File to Create a File BVS Repository](#)” on page 224

After you create the repository, you can use some of these files to perform other tasks using the `NdRomAdminUtil` utility, see “[The NdRomAdminUtil Utility Commands](#)” on page 162.

After you have created the File BVS repository with private workspaces, you may want to verify the versioning commands, see “[Verifying Blaze Advisor Versioning Commands](#)” on page 230.

Creating a Local or Private Workspace

If you need to access the contents of a versioned repository with local or private workspaces to work on its contents, you need to create a new workspace using the IDE or the utility. The types of workspaces you can create correspond to the repository type you created and include, File, Database (RDBMS), or MongoDB(NoSQL). The method you choose is based on personal preference.

When you create a workspace using the Blaze Advisor IDE, you also create a connection that can be used to access the workspace in the IDE. When you create a workspace using the `NdRomAdminUtil` utility, you need to create a connection in the Blaze Advisor IDE. See “[Creating a New Blaze Advisor Workspace or Repository Connection](#)” on page 58.

If you created a File BVS repository with a shared workspace or a MongoDB BVS repository with a shared workspace, the location of the repository and the workspace are considered to be the same. Instead of creating a new workspace for this type of repository, you create a new connection. See “[Connecting to a Blaze Advisor Repository](#)” in *DevelopingRuleProjects.pdf*.

This section includes these topics:

- “[Creating a Workspace in the Blaze Advisor IDE](#)” on page 227
- “[Creating a Blaze Advisor Workspace Using the Utility](#)” on page 228

To create a workspace to connect to a MongoDB repository, see “[Creating or Removing a MongoDB or File Workspace](#)” on page 365.

Creating a Workspace in the Blaze Advisor IDE

If you want to access a versioned Blaze Advisor repository to edit or update its contents, you need to create a Blaze Advisor workspace to populate the workspace with a copy of the repository contents, modify the projects and project files, and check in your changes in the Blaze Advisor.

To create a workspace

- 1 Select **Repository > New Workspace**.

Alternatively, from either the **File** menu or the Repository Explorer, select **New > Other** and expand the **Blaze Advisor** folder to choose **New Workspace**.

- 2 Select the repository type for the workspace you want to create.

You can select from the following repository types:

- File Repository (BVS private workspace)
- File Repository (CVS versioning)
- File Repository (Subversion versioning)
- Database Repository (BVS private workspace)
- MongoDB Repository (BVS private workspace)
- MongoDB Repository (BVS private local file workspace)



Note If you are using a ClearCase repository, you need to create your workspace in ClearCase, see "["Creating ClearCase Workspace Connections" on page 275](#)".

- 3 Enter a connection name for your workspace.

After the workspace is created, the connection name you entered here is displayed in the connection lists.

- 4 Click **Next**.

- 5 On the Connection Parameters page, enter the connection settings for the repository and the location information of the workspace.

- 6 (Optional) You can select the **Save user and password** check box to use a link to quickly and easily connect to the workspace or repository in Blaze Advisor. See "["Saving User Name and Passwords" on page 43](#)".

- 7 Click **Finish**.

You connect to the workspace by opening a project in Blaze Advisor. See "["Connecting to a Blaze Advisor Workspace or Repository" on page 52](#)".

After you connect to the workspace, you may need to explicitly update your workspace to see the latest versions of all files.

If you need to create a new connection, see "["Creating a New Blaze Advisor Workspace or Repository Connection" on page 58](#)".

Creating a Blaze Advisor Workspace Using the Utility

If you want to access a versioned repository with a private or local workspace, to modify its contents, you can create a Blaze Advisor workspace using the `NdRomAdminUtil` utility. Before you can use the utility, you need to write a connection configuration file for the workspace. Before you can run the utility, you need to write one or more connection configuration files. See:

- “[About Writing Files for Creating a Blaze Advisor Repository or Workspace Using the Utility](#)” on page 168
- “[Writing a Connection Configuration File for a File BVS Repository](#)” on page 221
- “[Writing a Connection Configuration File for a Workspace](#)” on page 228
- “[Writing a Batch File to Create a Workspace](#)” on page 229

After you create the repository, you can use some of these files to perform other tasks using the `NdRomAdminUtil` utility, see “[The NdRomAdminUtil Utility Commands](#)” on page 162.

When you run the utility, using either a batch file or by entering the commands in a command prompt, you reference the path to the workspace connection configuration file and the repository connection configuration file you used earlier to create the repository. When the workspace is created, the `com.blazesoftware.repository_config.cfg` file that is added to the root of the workspace directory contains repository connection information. In the Blaze Advisor IDE, use the New Repository Connection wizard to create a connection to the workspace or repository you just created.

By default, a workspace file is a File type, however you can create a workspace that is another Blaze Advisor type, see “[Writing a Connection Configuration File for a Private or a Local Workspace](#)” on page 185.

After creating the workspace, you may want to perform some administrative tasks using the utility. To perform some of these tasks you need to reference the workspace connection configuration file. See “[Importing or Exporting Projects and Workspace Contents Using the Utility](#)” on page 65.

Writing a Connection Configuration File for a Workspace

If you have created a BVS repository (File or Database) with a private workspace or a file CVS, Subversion or Clearcase repository with a local workspace, each time you want to create an additional workspace using the `NdRomAdminUtil` utility, you need to write or edit an existing workspace connection configuration file.

You can create the file using a text editor and save the file with a .cfg extension. For information about the tags and values in the following example, see “[Writing a Connection Configuration File for a Private or a Local Workspace](#)” on page 185.

The following is an example of a connection configuration file for a workspace:

```
<?xml version="1.0" encoding="UTF-8" ?>  
  
<RepositoryConnection>  
  <Factory>
```

```

com.blazesoft.repository.file.NdFileRepositoryConnection
</Factory>
<!--The workspace location must be edited for each workspace-->
<RepositoryFolder> C:\RepoWksp\BVSRepo_Wksp\BVSRepo3563_Dir
</RepositoryFolder>
<RepositoryName>BVSRepo3563_Dir </RepositoryName>
<User>user1</User>
<Password> pswd </Password>
</RepositoryConnection>

```

Writing a Batch File to Create a Workspace

To create a workspace using the utility, you can use a batch file or command prompt. In the file or prompt, you add the following command, argument, and argument values. See “[Running the NdRomAdminUtil Utility](#)” on page 166.

Command, Argument, or Option	Description
createWorkspace	Command to create a workspace.
-repositoryConnection	Argument takes the path to the location of the repository connection configuration file (.cfg). See “Repository Connection Configuration File Examples” on page 171 . This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
-workspaceConnection	Argument takes the path to the location of the Workspace connection configuration file (.cfg). See “Writing a Connection Configuration File for a Private or a Local Workspace” on page 185 . This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
-verbose	Option prints out all details messages into the command console.

Example of a batch file to create a File workspace using pathnames:

```

java com.blazesoft.template.repository.admin.NdRomAdminUtil createWorkspace
-repositoryConnection "c:/Repo_Connections/ConnectionFiles/
DefaultVerRepo.cfg"
-workspaceConnection "c:/Repo_Connections/ConnectionFiles/
DefaultVerWksp.cfg"
-verbose

```

Removing a Workspace Using the Utility

You can remove a workspace using the `NdRomAdminUtil` utility. You can use the connection configuration file for the repository and the workspace connection configuration file.

- [“Writing a Connection Configuration File for a File BVS Repository” on page 221](#)
- [“Writing a Connection Configuration File for a Workspace” on page 228](#)

- ["Writing a Batch File to Remove a Workspace" on page 230](#)

Writing a Batch File to Remove a Workspace

To remove a workspace using the utility, you can use a batch file or command prompt. In the file or prompt, you add the following command, argument, and argument values. See ["Running the NdRomAdminUtil Utility" on page 166](#).

Command, Argument, or Option	Description
removeWorkspace	Command to remove a workspace.
-workSpaceConnection	Argument takes the path to the location of the Workspace connection configuration file (.cfg). See "Writing a Connection Configuration File for a Private or a Local Workspace" on page 185 . This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
-verbose	Option prints out all details messages into the command console.

Example of a batch file to remove a File workspace using pathnames:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil removeWorkspace  
-repositoryConnection "c:/Repo_Connections/ConnectionFiles/  
DefaultVerRepo.cfg"  
-workspaceConnection "c:/Repo_Connections/ConnectionFiles/  
DefaultVerWksp.cfg"  
-verbose
```

Creating BVS Workspace Connections

You need to create connections for your users in the Blaze Advisor IDE or a rule maintenance application.

- You explicitly create a connection to a BVS workspace in the Blaze Advisor IDE. See ["Creating a New Blaze Advisor Workspace or Repository Connection" on page 58](#).
- When you generate an RMA, you can decide whether to use the current workspace or create one specifically for use with the RMA. For more information see, "Selecting Workspace Options" in *DevelopingRuleMaintenanceApplications.pdf*.

Verifying Blaze Advisor Versioning Commands

After you have created a repository with a versioning service, you can verify that your versioning commands are working correctly by testing the behavior of the commands in Blaze Advisor. The following table lists the versioning commands available when your users open a project from a versioned repository.

Versioning Commands	To verify that this command is working correctly
Check Out	<p>Select the project item in the Project Explorer and right-click and select Team > Check Out from the context menu.</p> <p>The Locked By Me icon appears next to the item that was checked out.</p> <p>If the project item that you want to check out is a folder, the Check Out window opens allowing you to select one or more items from the folder to check out.</p> <p>The Check Out command is not available to you if you attempt to check out an item locked by another user.</p> <p>See:</p> <p>“Checking Out One or More Items in the Blaze Advisor IDE” in <i>DevelopingRuleProjects.pdf</i>.</p> <p>“Checking Out One or More Items Using the Utility” on page 85.</p>
Check In	<p>Select the project item in the Project Explorer and right-click to select Team > Check In from the context menu. A timestamp and a user name are displayed beside each entity that was checked in.</p> <p>The Check In window opens where you can select one or more items to check in, enter a comment and click OK. The Locked By Me icon disappears after you check in the project item. This item is now available for other users to check out.</p> <p>See:</p> <p>“Checking In New or Existing Items in the Blaze Advisor IDE” in <i>DevelopingRuleProjects.pdf</i>.</p> <p>“Checking In New or Existing Items Using the Utility” on page 84.</p>
Cancel Check Out	<p>Select the project item in the Project Explorer and right-click to select Team > Cancel Check Out from the context menu.</p> <p>You are asked to confirm the cancel check out action, click OK.</p> <p>The Locked By Me icon disappears after you use the Cancel Check Out command. This project item is now available for other users to check out.</p> <p>See:</p> <p>“Canceling a Check Out for One or More Items in the Blaze Advisor IDE” in <i>DevelopingRuleProjects.pdf</i>.</p> <p>“Canceling the Check Out for One or More Items Using the Utility” on page 86.</p>
Update	<p>If you have a Blaze Advisor private or local workspace, you can use the Team > Update command from the context menu to update your Eclipse workspace with the latest versions of the files that have been checked into the repository.</p> <p>If the latest versions of the files are already in the workspace, the files are not replaced.</p>
Refresh	<p>If you have a File BVS repository with a shared workspace, you can use the Team > Refresh command from the context menu to update the contents of your project with the latest versions of the files that have been saved or checked into the repository. You can use the Refresh command to track the changes other users make to the project files.</p>

Versioning Commands	To verify that this command is working correctly
Status Check/Update	<p>If you are using a private or local workspace, you can highlight a project item and use the Team > Status Check/Update from the short-cut menu to check for differences between the version of the file in your Eclipse workspace and the version in the repository.</p> <p>The Status Check/Update window opens if the system detects a difference between the versions. You can choose to update the file so that you are working with the current version in your workspace. Locally created files cannot be updated until they are checked into the repository.</p> <p>See:</p> <ul style="list-style-type: none"> ■ “Checking the Status of Workspace Files in the Blaze Advisor IDE” in <i>DevelopingRuleProjects.pdf</i> ■ “Checking the Status of Workspace Files Using the Utility” on page 90.
Restore	<p>You can highlight a project item in the Project Explorer that has been logically deleted and select Team > Restore from the short-cut menu.</p> <p>This command restores a file to the project namespace, you can view its version history information and use it in your project. See:</p> <ul style="list-style-type: none"> ■ “Restoring Deleted Items in the Blaze Advisor IDE” in <i>DevelopingRuleProjects.pdf</i> ■ “Restoring Logically Deleted Items” on page 88.
Get Version <number>	<p>Open a project item in an editor and click the History tab. Right-click a version of the item you want to view in the Version History section of the History tab and select the Get Version <number> command</p> <p>Obtains the version you want to view and displays it in an editor.</p> <p>See:</p> <ul style="list-style-type: none"> ■ “Viewing a Previous Version of an Item in the Blaze Advisor IDE” in <i>DevelopingRuleProjects.pdf</i>. ■ “Viewing Item History Using the Utility” on page 89.
Promote Version	<p>Check out and open a project item in an editor and click the History tab. Right-click a version of the item you want to view in the Version History section of the History tab and select the Promote Version <number> command. See:</p> <ul style="list-style-type: none"> ■ “Promoting a Previous Version of an Item in the Blaze Advisor IDE” in <i>DevelopingRuleProjects.pdf</i> ■ “Promoting an Item Using the Utility” on page 91.

Converting a Blaze Advisor Repository to Using a Blaze Versioning (BVS)

There are several ways you can *convert* the contents of a Blaze Advisor repository to and from a repository with BVS versioning. The Import and Export commands in the Blaze Advisor IDE or the utility allow you to move the source workspace contents to a target BVS workspace where you can check in the files. For example, if you have a File repository with no versioning and you want to use a File BVS repository with private workspaces, you can export the source workspace contents to the target File BVS workspace and check in the files. From that point, you can connect to the File BVS workspace to continue working on the files. Likewise, if you have a Database repository

and you want to use a File BVS repository with private workspaces, you can use the same commands in the Blaze Advisor IDE or the utility command to convert the contents created in the Database repository to use a File BVS workspace.

Connections to both the source and target workspace must exist for you to convert the workspace contents successfully. In Blaze Advisor, to export a workspace, you can use either the File > Export and expand the Blaze Advisor Repository Administration folder and Workspace folders and select, To another Blaze Advisor workspace or to import a workspace, you can use File > Import and expand the Blaze Advisor and Workspace folders to select, From another Blaze Advisor workspace, as long as you have access to both the source and target workspace or repository connections. If you use the utility, you use the `exportImportWorkspace` command and reference the source and target workspaces.

If the target workspace is a BVS shared workspace you export the contents of the source workspace to the same physical directory location as the target repository. If the target is a BVS private workspace, you move the contents of the source workspace to the directory location of the private workspace. In both cases because you are using the workspaces to convert the container type you are working in, you need to check in the files after export.

If you have a File CVS repository and you want to convert to use a File BVS repository with a private workspace, assuming that both the source and target workspaces are file type, you can move the contents from your source workspace to the target workspace and check in the changes. If the workspace types are not both file, you may need to convert the workspace(s) to be a file type before proceeding with the conversion. Please note that the version history is not persisted to the target repository connection.

You can use one of these methods to move your workspace contents:

- “Importing a Blaze Advisor Contents from Another Blaze Advisor Workspace” in *DevelopingRuleProjects.pdf*
- “Exporting a Blaze Advisor Contents to Another Blaze Advisor Workspace” in *DevelopingRuleProjects.pdf*
- “[Exporting and Importing a Workspace Using the Utility](#)” on page 75

CHAPTER 7

Implementing a CVS Repository

You can create a File CVS repository in FICO® Blaze Advisor® decision rules management system if you have a CVS server installed on a network machine. The topics in this chapter discuss how you can implement a File CVS repository to manage Blaze Advisor rule projects. By default, each user connected to the CVS repository checks out files to their local file workspace in order to edit or update their contents before checking them back into the repository on the CVS server.

The topics in this chapter include:

- “[Setting Up a CVS Server](#)” on page 236
- “[Local Workspaces](#)” on page 238
- “[Creating a CVS Repository](#)” on page 239
- “[Converting a Repository to and from a File CVS Repository](#)” on page 247

You cannot create a CVS repository with a connection to a CVSNT server in the Blaze Advisor IDE. However you can use the `NdRomAdminUtil` utility. For an example of the files you need to create for using a CVSNT server, see “[Creating a CVSNT Repository](#)” on page 248.

If you already have a CVS server installed, you can create a CVS repository using one of these methods:

- “[Creating a CVS Repository in Blaze Advisor](#)” on page 239
- “[Creating a CVS Repository Using the Utility](#)” on page 242

If you need to create a new connection to an existing workspace or you need to create a new workspace for your CVS Repository, see “[Creating Workspaces and Workspace Connections for a CVS Repository](#)” on page 246.

The File CVS Repository type uses the CVS library contained in `javacvs.jar` installed with Blaze Advisor to connect to your CVS server. You do not need to install a separate CVS client.

If you want to add additional access controls to your repository, see “[Implementing an Authorization Manager](#)” on page 389.

If you plan to check in items outside the Blaze Advisor context, see “[Using the External SCM Versioning System Outside the Blaze Advisor Context](#)” on page 284.



Note You cannot use the same CVS workspace files on UNIX and Windows platforms without first converting the workspace files because of line ending issues. A workspace

checked out in Windows contains files with \r\n line endings and a workspace checked out on UNIX contains files with \n line endings. You would need to manually change the line endings prior to using the workspace files on the other platform.

Setting Up a CVS Server

If you and your development team are familiar with using CVS, you will find that many of the same commands for managing and tracking revisions are available when using a CVS repository. Blaze Advisor supports CVS server versions 1.11.17 and 1.11.22.

This section includes topics about:

- ["Verifying Your CVS Server Installation" on page 236](#)
- ["Creating a Repository Directory on a CVS Server" on page 236](#)
- ["Using a Password-authenticating Server" on page 237](#)

Verifying Your CVS Server Installation

You need to have the CVS server installed on a network machine to create a Blaze Advisor CVS repository with versioning. If a CVS server is not currently installed, your system administrators need to get a version that is compatible with the network machine and is supported by Blaze Advisor. Blaze Advisor supports CVS server versions 1.11.17 and 1.11.22.

After your system administrators have installed a CVS server or if you have an existing CVS server, ask your system administrators to set up a repository directory for your project. You need to reference the CVS repository directory path when connecting to a repository and workspace using a command prompt or the Blaze Advisor IDE.

After the CVS server is installed, for security reasons you may want your system administrators to explicitly add user names to a Unix group. Explicitly adding users to a group prevents the possibility of unauthorized users accessing the system. When using CVS versioning, you use the password authenticated connection method supported by Blaze Advisor, see ["Using a Password-authenticating Server" on page 237](#).

Creating a Repository Directory on a CVS Server

Once CVS is installed on your server machine, you can prepare a directory location where you will create the repository or repositories using Blaze Advisor.

- ["Creating a Repository Directory" on page 236](#)
- ["Creating a Module in the Repository Directory" on page 237](#)
- ["Creating a File CVS Repository Branch" on page 237](#)

Creating a Repository Directory

Before you can create a *repository directory* location on the CVS server machine, you need to be certain you have write permission to the server location. The repository directory location is where you can create one or more File CVS repositories. You may need to

check with the system administrator to confirm that you have write permission for the directory location you want to use. In some cases, you may need to login as the system administrator to create the repository directory location. Once you have write permission, you can use a command prompt and enter commands to create a repository directory location. For example:

```
cvs -d /usr/local/<path to the repository directory> init
```

where `/usr/local/` is the path where you want the repository directory to be located. The `-d` specifies the `CVSROOT` is added to this location. The `init` command creates the repository directory.



Note The `CVSROOT` you create is not the same as the environment variable for the `$CVSROOT`. In this case, the `CVSROOT` is the administrative subdirectory of the repository directory.

Creating a Module in the Repository Directory

When you create a File CVS repository using the Blaze Advisor IDE or the `NdRomAdminUtil` utility, you are in effect creating a module in the repository directory you created on the CVS server. You can create one or more repository modules in this location as long as you have the appropriate permissions. You need to set the user permissions for that repository directory and any modules so that your users can connect to the repositories. See "[Using a Password-authenticating Server](#)" on page 237.

Creating a File CVS Repository Branch

You can branch the repository directory to develop projects in separate repositories. The main benefit to branching File CVS repositories is that you can develop projects in parallel. It is possible to create a repository with a set of projects and files and another identical repository with a set of project and files that will have additional bug fixes and features. However, please note, no merging is supported. You need to use the `tag` command in CVS to create a branch. If you want to create a File CVS Repository using an existing branch, you enter the name of the branch when you create the repository in the Blaze Advisor IDE or the `NdRomAdminUtil` command-line utility.

Using a Password-authenticating Server

The password-authenticating server or `pserver` method is the only network access method currently supported by Blaze Advisor. The `pserver` connection method can be used to provide different permission levels for accessing your repository based on user names and passwords.

Some versions of CVS include the `pserver` connection method in their installations. If the `pserver` connection method is not available on your CVS server, you need to ask your system administrators to add it by modifying the `inetd` configuration files in `/etc/services` and `/etc/inetd.conf` files.

When a remote CVS client uses the `pserver` method to connect to a repository, the client is actually connecting a specific port number on the server machine. The default port number is 2401.

The CVS server itself does not wait for connection requests. Instead you have your system administrators set up the Unix `inetd` (Internet daemon) program to listen on port 2401 and instruct it to start up the CVS server and connect it to the incoming client when it receives a connection request.

Using Access Permissions Supported by Blaze Advisor

If your CVS server is configured to use the pserver method, you can ask your system administrators to specify the following levels of access for your users:

User Name and Password Access

You can ask your system administrators to create a `passwd` file in the CVSROOT directory of your repository using the user names and passwords that you supply. The passwords should be independent from the one used for system login.

If a user attempts a pserver connection with a user name and password that does not appear in the `passwd` file, the CVS default behavior is to check to see if the user name and password exist in the `/etc/passwd` file. If the user name and password exists in this `passwd` file then CVS grants access.

This means that you do not necessarily need to create a separate `passwd` file for regular system users to gain access to the repository. However, if you do not have separate passwords for connecting to the system and connecting to the repository, this represents a potential security hole. We recommend that you require your users to enter a different password depending on the tasks they want to perform and ask your system administrators to maintain separate `passwd` files.

We recommend that you ask your system administrator to use encrypted passwords. Your system administrator can use the Unix algorithm, `crypt()` method to encrypt passwords from text input and paste these passwords into the `passwd` file.

Project-specific Access

In some cases you may want the CVS server to grant users access to specific project files within a repository. Your system administrators can provide access to project files by creating project-specific accounts and mapping these accounts to the `etc\passwd` file and the `CVSROOT\passwd` file.

Because the CVS server checks the user name of the system account, your system administrators need to make sure that only the appropriate users and groups can write to the relevant parts of the repository by explicitly stating the project name in the `passwd` file. As an added safeguard, CVS records changes to project files using the CVS user name so that you can check on who is responsible for a given change.

Local Workspaces

A local workspace allows users to control how and when changes made to files, are available to others and to control when files changed by others, are made available to them. In this regard, local workspaces and the BVS private workspaces are similar. When you connect to a CVS, a Subversion, or ClearCase repository, you are establishing a connection between the repository on the server and the local workspace on your

machine. The login information you provide is the user name and password for access the server machine.

If you are connected to a CVS server machine, a CVS folder is added at the directory and folder levels in the local workspace to manage the version history of the files. The CVS folder contains the Entries, Root, and Repository files used by CVS to track the version history of files. If you are connected to a Subversion or ClearCase server, similar files and folders may be added to track the version history of files.

If you are connected to another source control management server machine, you connect to the view location for your workspace.

Users can perform tasks in a local workspace including:

- Check out files,
- Check in any changes to files
- Obtain updates
- Obtain version history
- Retrieve item status for files

Each user connected to the CVS or SCM repository has their own local workspace and the contents and structure of each local workspace can differ. The structure of a CVS or SCM local workspace is similar to a BVS private workspace, see "["File Structure for a Private or Local Workspace" on page 212](#)".

For general information on about repositories and workspaces, see "["The Blaze Advisor Repositories and Workspaces" on page 21](#)".

Creating a CVS Repository

If you have the CVS server installed on a machine and the CVS client on your machine, you can create a repository using one of the following methods:

- ["Creating a CVS Repository in Blaze Advisor" on page 239](#)
- ["Creating a CVS Repository Using the Utility" on page 242](#)

If your CVS server is not able to handle multiple commands, for example, if you have a CVSNT server, you need to create the repository using the `NdRomAdminUtil` utility instead of using the New Repository wizard in the Blaze Advisor IDE. When you write the connection configuration file, you add a multiple command option tag. See "["Creating a CVSNT Repository" on page 248](#)".

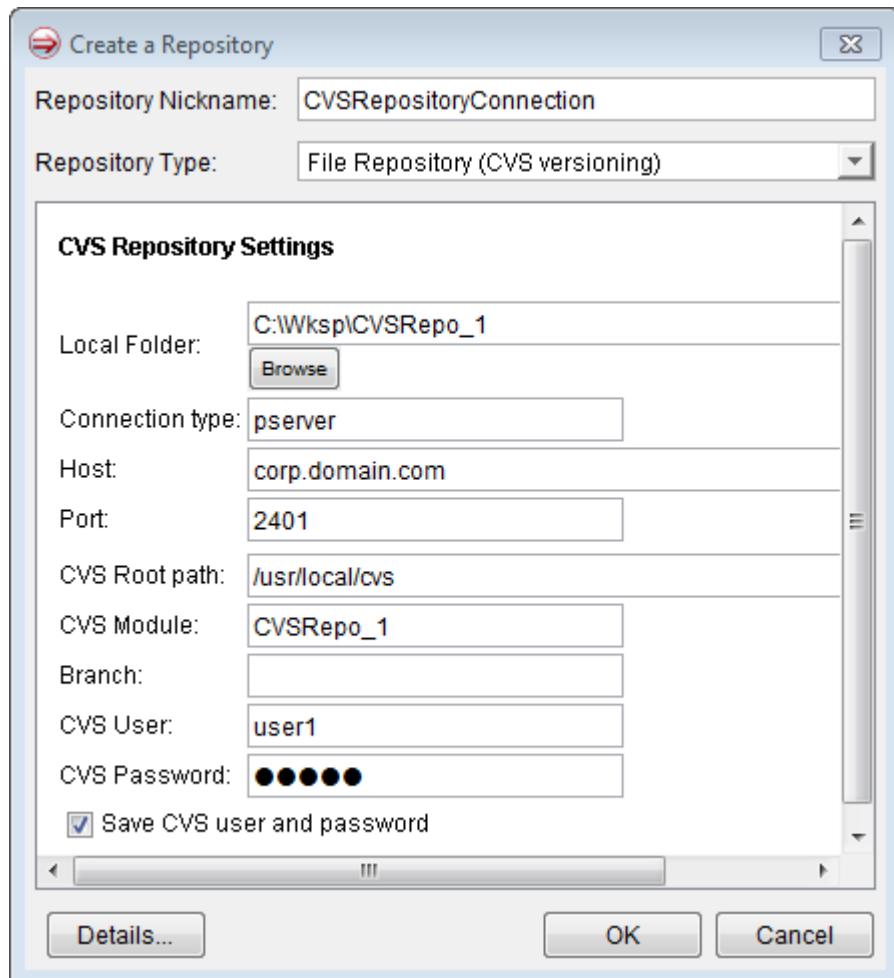
After you have created a CVS repository, you can verify the versioning commands in the Blaze Advisor IDE, see "["Verifying Blaze Advisor Versioning Commands" on page 230](#)".

Creating a CVS Repository in Blaze Advisor

You can create a CVS repository using the Blaze Advisor IDE after you have the CVS server installed on a network machine and a module created for your repository. Below is an example of the type of information you need to enter for the connection parameters.



Important To avoid any creation errors, when entering values into the fields of the Repository Connection Parameters page, leave no trailing spaces.



To create a CVS repository

- 1 Select **Repository > New Repository**.
- 2 Select a File Repository (CVS versioning) for the **Repository Type**
- 3 Enter the following information to connect to the machine that is hosting the server:

Repository Type Connection Parameter	Description
Local Folder	<p>The path to the local workspace folder for your File CVS repository connection.</p> <p>Important The workspace folder name you enter in the Local Folder field must be identical to the name you enter in the CVS Module field.</p>
Connection type	<p>pserver</p> <p>This is the CVS connection type currently supported by Blaze Advisor.</p>
Host	Host machine name
Port	2401 is the default port number.
CVS Root path	The UNIX root path that contains the repository directory on the CVS server.
CVS Module	<p>The name of the repository module you want to create in the CVS Root path.</p> <p>Important The CVS Module name and the name of the local workspace folder must be identical.</p>
Branch	<p>If you want to create a repository on a branch on the CVS server, enter the path.</p> <p>Note Blaze Advisor does not support merging.</p>
CVS User	User name for the CVS server.
CVS Password	Password for the CVS server.

- 4 (Optional) Select the **Save CVS user and password** option to save the user name and password in the connection instance file and the repository configuration file. See ["Saving User Name and Passwords" on page 43](#).

After you exit the wizard, you need to explicitly connect to a workspace using the Blaze Advisor IDE. See ["Connecting to a Blaze Advisor Workspace or Repository" on page 52](#).

After you create the repository, proceed with one or more of the following tasks:

- If you need to create additional workspaces for your users, see ["Creating a Local or Private Workspace" on page 226](#).
- If you want to create a new workspace connection to the CVS repository, see ["Creating a New Blaze Advisor Workspace or Repository Connection" on page 58](#).
- After you have created the repository with a local workspace, you may want to verify the versioning commands, see ["Verifying Blaze Advisor Versioning Commands" on page 230](#).

Creating a CVS Repository Using the Utility

You can create a File Repository (CVS versioning) by using the `NdRomAdminUtil` utility. Before you can run the utility to create the repository, you need to write two connection configuration files and a configuration file. You also need to write a connection configuration file for the local workspace.

- “[About Writing Files for Creating a Blaze Advisor Repository or Workspace Using the Utility](#)” on page 168
- “[Writing the Connection Configuration File for an Admin Repository](#)” on page 33
- “[Writing the Connection Configuration File for the File CVS Repository](#)” on page 242
- “[Writing the Repository Configuration File for the File CVS Repository](#)” on page 244
- “[Writing a Batch File for Creating the File CVS Repository](#)” on page 245

Writing the Connection Configuration File for the File CVS Repository

You can write a connection configuration file (`.cfg`) for a File repository with CVS versioning that connect to either a CVS or CVSNT server. This connection file needs to contain the structure and the values described in the table below.

You can use a local workspace with a CVS repository. In the repository configuration file, when you specify that you are using a CVS repository, you automatically use a local workspace. You need to write a separate connection configuration file for the local workspace and run the utility a second time. See “[Writing a Connection Configuration File for a Private or a Local Workspace](#)” on page 185.



Note For reasons of brevity, a pair of element tags will be represented by the following notation, `<example/>`.

Tag	Description
<code><?xml version="1.0" encoding="UTF-8" ?></code>	Processing information that contains the XML version and character-encoding used.
<code><Repository Connection/></code>	Used when you need to create a connection for a repository with versioning. All other tags in this table are nested within this tag.
<code><Factory/></code>	Used when you want to create an instance of a class. To create an instance of a connection for CVS versioning, you use the fully qualified class name: <code>com.blazesoft.repository.file.NdFileCVSServerConnection</code> .

Tag	Description
<PersistCredentials/>	<p>(Optional) Used to save the username and password to the connection instance file in the Admin Repository.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ■ false -- do not store connection credentials (default) If you choose not to store the connection information, each time you connect to the repository in the Blaze Advisor IDE, you need to manually enter your user name and password. ■ true -- store the connection credentials <p>See “Saving User Names and Passwords Using the Utility” on page 183.</p>
<DisableMultipleCommands/>	<p>Used if you have a CVS server, such as a CVSNT server, that cannot handle multiple commands. By default, the value for the tag is false.</p> <p>If you have a CVSNT server, the value is true. See “Creating a CVSNT Repository” on page 248.</p>
<Type/>	<p>Used to specify the connection type. The connection type currently supported by Blaze Advisor is pserver.</p>
<Host/>	<p>Used to specify the name of the host machine. This is the machine that has the CVS server installed.</p>
<Name/>	<p>Used to specify the name of the repository module that houses the CVS repository.</p>
<Port/>	<p>Used to specify the port number. The default port number is 2401.</p>
<Path/>	<p>Used to specify the path to the module.</p>
<RepositoryName/>	<p>The name of the repository directory on your local machine. This value must be the same as the repository module specified in the <Name/> tag.</p>
<User/>	<p>Used to specify the user name for the connection to the CVS server.</p>
<Password/>	<p>Used to specify the password for the connection to the CVS server.</p>
<Branch/>	<p>(Optional) If branching is used, the name of the branch.</p>

After you have written the connection file for your CVS repository, you need to write or edit a configuration file for a local workspace. See “[Writing a Connection Configuration File for a Private or a Local Workspace](#)” on page 185.

You run the `NdRomAdminUtil` utility twice to create a repository with a local workspace. First you run the utility to create the repository and then you run it a second time to create the workspace. Each time you run the utility, you must use the correct arguments and options for each task.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<RepositoryConnection>
    <Factory> com.blazesoftware.repository.file.NdFileCVSServerConnection
    </Factory>
    <PersistCredentials> true </PersistCredentials>
    <DisableMultipleCommands> false </DisableMultipleCommands>
    <Type>pserver </Type>
    <Host>boxster </Host>
    <Port>2401 </Port>
    <Path>/projects/branches/repository2 </Path>
    <Branch>repo3xbranch</Branch>
    <Name>CVSRepo100 </Name>
    <User>user1 </User>
    <Password>user1 </Password>
    <RepositoryName>CVSRepo100</RepositoryName>
</RepositoryConnection>
```

Writing the Repository Configuration File for the File CVS Repository

You can use this repository configuration file when creating a File repository with CVS versioning.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<RomConfig>
    <AllowUnusedValues> true </AllowUnusedValues>
    <RomConnectionManagerConfig>
        <ConnectionMode> 0 </ConnectionMode>
        <RomConnectionManagerFactory>
            <JavaName>
                com.blazesoftware.template.repository.impl.NdDefaultRomConnectionManager
            </JavaName>
        </RomConnectionManagerFactory>
    </RomConnectionManagerConfig>
    <RomQueryManagerConfig>
        <RomQueryManagerFactory>
            <JavaName>
                com.blazesoftware.template.repository.query.NdRomDefaultQueryManager
            </JavaName>
        </RomQueryManagerFactory>
    </RomQueryManagerConfig>
    <RomSchemaManagerConfig>
        <RomSchemaManagerFactory>
            <JavaName>
                com.blazesoftware.template.repository.impl.NdDefaultRomSchemaManager
            </JavaName>
        </RomSchemaManagerFactory>
    </RomSchemaManagerConfig>
```

```

</RomSchemaManagerConfig>
<RepositoryConfig>
    <AtticRepository> true </AtticRepository>
    <RepositoryVersionManagerConfig>
        <RepositoryVersionManagerFactory>
            <JavaName>
                com.blazesoftware.repository.file.NdFileCVSWorkspaceVersionManager
            </JavaName>
        </RepositoryVersionManagerFactory>
        <RepositoryVersionParserFactory>
            <JavaName>
                com.blazesoftware.repository.file.NdDefaultCVSRepositoryVersionResult
                    Parser
            </JavaName>
        </RepositoryVersionParserFactory>
        <PrivateWorkspaceSupported> true </PrivateWorkspaceSupported>
    </RepositoryVersionManagerConfig>
</RepositoryConfig>
...
</RomConfig>

```

Writing a Batch File for Creating the File CVS Repository

You run the `NdRomAdminUtil` utility twice to create a File CVS repository. First you run the utility to create the repository and then you run it a second time to create the workspace. Each time you run the utility, you must use the correct arguments and options for each task. See ["Creating a Blaze Advisor Workspace Using the Utility" on page 228](#).

Command, Argument, or Option	Description
<code>createRepository</code>	Command to create a repository
<code>-repositoryConnection</code>	Argument takes the path to the location of the repository connection configuration file (.cfg). See "Repository Connection Configuration File Examples" on page 171 . This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
<code>-workspaceConnection</code> Note Use this argument to create a workspace for a BVS Repository with private workspaces, File CVS Repository File Repository (Subversion versioning) or a File Repository (ClearCase versioning) using an IBM Rational ClearCase server connection.	Argument takes the path to the location of the workspace connection configuration file (.cfg) See "Writing a Connection Configuration File for a Private or a Local Workspace" on page 185 . This file contains information such as the repository name, the workspace connection name, the user name and password for the connection, and the path to the workspace.
<code>-systemconnection</code>	Argument takes the path to the location of the system Folder connection configuration file (.cfg). See "Repository Connection Configuration File Examples" on page 171 . This file contains information such as the name of the repository which in this case is Admin Repository, the location of the Admin Repository and the user name.

Command, Argument, or Option	Description
--configuration	Argument takes the path to the location of the configuration file(.cfg). See " Writing a Repository Configuration File " on page 191.
-verbose	Option prints out all details messages into the command console.
-m	Option to add a comment when creating a repository. Note Use double quotes around your comment text
-debug	Option prints out tracing information into the command console.

Example of a batch file to create a File BVS repository using pathnames:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil  
createRepository  
-repositoryConnection C:/Repo_Connections/ConnectionFiles/DefaultRepo.cfg  
-systemConnection C:/Repo_Connections/ConnectionFiles/sysAdmin.cfg -  
configuration C:/Blaze/Advisor/lib/FileCVSRepository.cfg -verbose
```

After you have created the repository with a local workspace, you may want to verify the versioning commands, see "[Verifying Blaze Advisor Versioning Commands](#)" on page 230.

Creating Workspaces and Workspace Connections for a CVS Repository

If you want your users to connect to the CVS repository using the Blaze Advisor IDE, you can do one or more of the following:

- Create a new connection to an existing workspace
See "[Creating a New Blaze Advisor Workspace or Repository Connection](#)" on page 58.
- Create a new workspace
See "[Creating a Local or Private Workspace](#)" on page 226.
- When you generate an RMA, you can decide whether to use the current workspace or create one specifically for use with the RMA. For more information see, "Selecting Workspace Options" in *DevelopingRuleMaintenanceApplications.pdf*.

Converting a Repository to and from a File CVS Repository

There are several ways you can convert the contents of a repository to and from a File CVS repository in Blaze Advisor. When you use the commands in the Blaze Advisor IDE or the utility command to convert the source workspace contents, you are essentially importing or exporting the contents of the source workspace to use the new File CVS repository connection type. For example, if you have a File repository and you want to use a File CVS repository, you can export the source workspace contents to a target workspace created for a File CVS repository connection and check in the files. From that point, you can connect to the File CVS repository and continue working on the files in the

new workspace. Likewise, if you have a Database repository and you want to use a File CVS repository, you can use the same commands in the Blaze Advisor IDE or the utility command to convert your workspace created in the Database repository to use a File CVS repository connection and check in your files.

Both the source and target workspace must already exist for you to convert the workspace contents successfully. In the Blaze Advisor IDE, you can use either the File > Export or File > Import command depending on whether you have the source or target workspace open. If you have the source workspace open and you want to convert to another repository, you would use the File > Export > Export current workspace to another repository command or if you have the target workspace open, you would use the File > Import > Import from another repository command. If you use the utility, you use the `exportImportWorkspace` command and reference the source and target workspaces.

If you have a BVS repository with a shared or private workspace and you want to convert to use a File CVS repository with a local workspace, assuming that both the source and the target workspaces are of file type, you can move the contents from your source workspace to the target workspace and check in the changes. If the workspace types are not both file, you may need to convert the workspace(s) to be a file type before proceeding with the conversion. Please note that the version history is not persisted to the target repository connection.

You can use one of these methods to move your workspace contents:

- “Importing a Blaze Advisor Contents from Another Blaze Advisor Workspace” in *DevelopingRuleProjects.pdf*
- “Exporting a Blaze Advisor Contents to Another Blaze Advisor Workspace” in *DevelopingRuleProjects.pdf*
- “[Exporting and Importing a Workspace Using the Utility](#)” on page 75

Important

- You cannot use the Copy and Paste commands to transfer items from a File repository into a File CVS repository. You use one of the methods listed above to move the files to another workspace.
- After the conversion process is complete, you need to check in any project items in the target workspace to the repository.
- If you want to convert a BVS Database repository with a private workspace to a File CVS repository, and the private workspaces of the source repositories are File type, you can move the contents of the private workspace to the local workspace and check them in. The conversion in this case would be fairly straightforward. Please note that the version history of the is not persisted to the target repository connection.
- If you want to convert a BVS Database repository with a private workspace to a File CVS repository, and the private workspaces of the source repositories are not File type, you may need to make some changes to the contents of the private workspace before you can move the contents to the local workspace. Please note that the version history of the is not persisted to the target repository connection.

Creating a CVSNT Repository

You can create a CVSNT repository using the `NdRomAdminUtil` utility in the current version of Blaze Advisor. Currently you can not use the New Repository wizard in the Blaze Advisor IDE to create this repository type because a CVSNT repository connection requires the addition of the `<DisableMultipleCommands>` tag as a parameter. This tag takes a boolean value. By default value for most CVS repositories is *false*, However, for CVSNT repositories and any other repositories that cannot use multiple commands, the value must be set to *true*.

Prerequisites for Creating a CVSNT Repository

Prior to creating the CVSNT repository, you need to have CVSNT server installed on a machine. You also need to know the parameters for connecting to the machine, just as you would if you were connecting to a CVS repository. When you use the `NdRomAdminUtil` utility, you need to write two connection configuration files and a ROM configuration file.

Examples of all the configuration files you need to create are provided in these topics:

- ["Writing a Connection Configuration File for a CVSNT Repository" on page 248](#)
- ["Writing a Connection Configuration File for the system Folder" on page 249](#)
- ["Editing the Repository Configuration File" on page 249](#)

After you have created these files, you can write a batch file to run the `NdRomAdminUtil` utility. See ["Running a Batch File to Create a CVSNT Repository" on page 250](#).

Writing a Connection Configuration File for a CVSNT Repository

You write a connection configuration file that contains tags for the parameters to connect to a CVSNT server when you want to create a CVSNT repository using the `NdRomAdminUtil` utility. The connection parameter tags for a CVSNT repository are similar to those you use for a CVS repository. See ["File Repository with CVS Versioning" on page 177](#).

Example of a repository connection configuration file for a CVSNT repository:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<RepositoryConnection>
    <Factory> com.blazesoftware.repository.file.NdFileCVSServerConnection
    </Factory>
    <DisableMultipleCommands> true </DisableMultipleCommands>
    <Host> test001 </Host>
    <Name> New Repository </Name>
    <Path> /cvsrepo </Path>
    <RepositoryName> test2 </RepositoryName>
    <Type> pserver </Type>
    <User> tech1 </User>
    <Password>tech1</Password>
</RepositoryConnection>
```

Writing a Connection Configuration File for the system Folder

You write a connection configuration file for the system folder in the Admin Repository located in the /lib directory of your Blaze Advisor installation. A copy of the system folder is added to the CVSNT repository.

Example of a connection configuration file for the Admin Repository installed with Blaze Advisor:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RepositoryConnection>
    <Factory> com.blazesoftware.repository.file.NdFileRepositoryConnection
    </Factory>
    <Name> Admin </Name>
    <RepositoryFolder> C:/<ADVISOR_HOME>/lib/Admin Repository
    </RepositoryFolder>
    <User> user </User>
    <RepositoryName> Admin Repository </RepositoryName>
</RepositoryConnection>
```

Editing the Repository Configuration File

You need to edit a copy of the romConfig.cfg file that is provided in the <ADVISOR_HOME>/lib directory to include the CVS connection manager.

Example of an edited romConfig.cfg file. A generic version of this file is provided for your convenience at <ADVISOR_HOME>\lib. You edit this file when you need to create a repository of a given type or change the configuration of an existing repository using the NdRomAdminUtil utility. See “[Writing a Repository Configuration File](#)” on page 191.

```
<?xml version="1.0" encoding="UTF-8" ?>

<RomConfig>

    <RomQueryManagerConfig>
        <RomQueryManagerFactory>
            <JavaName>
com.blazesoftware.template.repository.query.NdRomDefaultQueryManager </
JavaName>
            </RomQueryManagerFactory>
        </RomQueryManagerConfig>

        <RomConnectionManagerConfig>
            <ConnectionMode> 0 </ConnectionMode>
            <RomConnectionManagerFactory>

                <JavaName>com.blazesoftware.template.repository.impl.
NdDefaultRomConnectionManager</JavaName>
            </RomConnectionManagerFactory>

        </RomConnectionManagerConfig>

        <RomSchemaManagerConfig>
            <RomSchemaManagerFactory>
```

```
<JavaName>
com.blazesoft.template.repository.impl.NdDefaultRomSchemaManager
</JavaName>
</RomSchemaManagerFactory>

</RomSchemaManagerConfig>

<RepositoryConfig>
<RepositoryVersionManagerConfig>
<PrivateWorkspaceSupported> true </PrivateWorkspaceSupported>
<RepositoryVersionManagerFactory>
<JavaName>
com.blazesoft.repository.file.NdFileCVSWorkspaceVersionManager
</JavaName>
</RepositoryVersionManagerFactory>
</RepositoryVersionManagerConfig>
</RepositoryConfig>

<AllowUnusedValues> true </AllowUnusedValues>

</RomConfig>
```

Running a Batch File to Create a CVSNT Repository

After you have written your configuration files, you can write a batch file to run the `NdRomAdminUtil` utility with the `createRepository` command and the arguments and options to create a CVSNT repository.

Example of a batch file used to run the `NdRomAdminUtil` utility.

```
<ADVISOR_HOME>/bin/setenv
java com.blazesoft.template.repository.admin.NdRomAdminUtil
createRepository -repositoryConnection "c:\ < Location of your repository
connection config file >\ACVSNTRepo.cfg" -systemConnection "<Location of
your admin connection config file > \admin.cfg" -configuration "<Location of
your ROM connection config file >\romCVSNTConfig.cfg" -verbose
```

It may take approximately 30 minutes or more for the creation process to be completed on a remote machine.

When the creation process is complete, you should see the following message:

```
Command <createRepository> is executed successfully.
```

After you have created the repository, you need to create a connection to the CVSNT repository using the Blaze Advisor IDE. You can create a local workspace at the same time. See “[Connecting to a Blaze Advisor Repository From a Rule Maintenance Application](#)” on page 55.

Here are some additional tips for creating a connection to a CVSNT repository in the Blaze Advisor IDE.

- Enter the Repository Connection Name you want to use and select File Repository (CVS versioning) as the Repository Type.

- Enter the connection parameter values for the CVSNT repository.
(Note that the Local Folder (workspace folder) and the CVS module name must be the same.)
- There is no connection parameter field corresponding to the <DisableMultipleCommands/> tag in this wizard.
- It may take more than a few minutes to create a repository connection to a remote machine.

After you create a connection in the Blaze Advisor IDE, you can begin using the workspace to create the entities and check them into the repository.

In future to create a new connection to the CVSNT repository, you use the same steps. You can also use the Update command on the repository toolbar in the Blaze Advisor IDE to populate the new workspace with the latest version of the project items.

CHAPTER 8

Implementing External SCM Repositories

FICO® Blaze Advisor® decision rules management system has several default repository types that connect to external SCM systems including:

- Subversion® (SVN) open source version control system
See “[Implementing a Subversion Repository](#)” on page 253.
- IBM® Rational® ClearCase® server version 8.0.1
See “[Implementing a ClearCase Repository](#)” on page 263.
- Concurrent Versions System (CVS) open source version control system.
See “[Implementing a CVS Repository](#)” on page 235.

Each user connected to an external SCM repository checks out files to their local file workspace to create, edit, or update files before checking files back into the repository on the server. See:

- “[Local Workspaces](#)” on page 238
- “[Creating a Local or Private Workspace](#)” on page 226

Blaze Advisor also has a built-in SCM system that allows connections to a shared workspace or private workspaces for File repositories and connections to a private workspace for Database repositories. The private workspaces are similar to the local workspaces used by external SCM systems. See “[Implementing a Repository With a Blaze Versioning Service](#)” on page 209.

If you have created a SCM repository using an earlier version of Blaze Advisor, see “[Migrating an SCM Repository Using IBM Rational ClearCase 7.1](#)” on page 280.



Important All workspaces must be ClearCase 8.0.1 workspaces connecting to ClearCase 8.0.1 server.

If you want to add additional access controls to your repository, see “[Implementing an Authorization Manager](#)” on page 389.

Implementing a Subversion Repository

You can create a file repository connecting to a server with Subversion (SVN) open source version control system using the Blaze Advisor IDE or the `NdRomAdminUtil` utility.

- “[Prerequisites for Creating a Subversion Repository in Blaze Advisor](#)” on page 254
- “[Creating a Subversion Repository in Blaze Advisor](#)” on page 255

- "Creating Workspaces or Workspace Connections for a Subversion Repository" on page 263

Prerequisites for Creating a Subversion Repository in Blaze Advisor

Before you can create a File repository (Subversion versioning) in Blaze Advisor, you need to ensure that the Subversion (SVN) is installed on a network machine and that you have the appropriate access permissions.

There are several prerequisites for creating a SVN repository. See "["Supported Versions of Subversion Clients and Servers" on page 254](#).

Supported Versions of Subversion Clients and Servers

If you want to use Blaze Advisor SVN repositories and the SVN server. You need to add the SVNkit and related jar files. The SVNkit is a pure Java third party toolkit that implements all Subversion features and provides a SVN client and APIs that are used to communicate with and store and track revisions in your Blaze Advisor SVN repository. Prior to creating an SVN Repository, you need to install the SVNkit plugin into Eclipse. By installing the plugin you install the necessary jar files and make them available to the Blaze Advisor IDE. See "["Installing the SVNkit Plugin" on page 255](#).



Tip We recommend that you use the same version TortoiseSVN client as your SVNkit to view your repository.

Versions of SVN server and SVNkit supported:

- Blaze Advisor supports SVN server versions 1.6.x and 1.7.x with SVNkit client 1.7.13. You can download the SVNkit 1.7.13 from the following third party Web site, see http://www.svnkit.com/org.tmatesoft.svn_1.7.13.standalone.nojna.zip.
- We recommend that you download the `org.tmatesoft.svn_1.7.13.standalone.nojna.zip`. The jar files you need are located in the `svnkit-1.7.13\lib` directory:
 - `antlr-runtime-3.4.jar`
 - `sequence-library-1.0.2.jar`
 - `sqljet-1.1.10.jar`
 - `svnkit-1.7.13.jar`
 - `svnkit-cli-1.7.13.jar`
 - `svnkit-javah16-1.7.13.jar`
 - `trilead-ssh2-1.0.0-build217.jar`

For more information about Subversion download. See <http://svnkit.com/download.php>.

Add the jar files from `svnkit-1.7.13\lib` to `<ADVISOR_HOME>/lib` and if the SVN jar file names are different, you need to edit the `setenv.bat` file located in `<ADVISOR_HOME>/bin`.

If you are deploying an RMA from a project created in an SVN repository, you need to ensure that the SVN jar files are added to the **WEB-INF/lib** directory. See "Adding Jar Files for Your Repository Type" in *DevelopingRuleMaintenanceApplications.pdf*.

Installing the SVNkit Plugin

Before you can use the Blaze Advisor IDE to create an SVN repository, you need to install the SVNkit plugin.

To install the SVNkit plugin

- 1 Select **Help > Install New Software**.
- 2 Click **Add** and in the Add Repository dialog, enter the following values:
 - a Enter a name for the installed plugin in the Name field.
 - b In the Location field, enter the URL for the SVNkit you want to install:
 - To use svnkit 1.7.x, enter `http://eclipse.svnkit.com/1.7.x/`
- 3 Click **OK**.
You should see the following plugins:
 - Core SVNkit Library (required)
 - Optional JNA Library (recommended)
- 4 Click **Select All** to begin installing the SVNkit plugin.
- 5 You may need to restart Eclipse to complete the installation.
- 6 You are now ready to create a Subversion Repository.

Creating a Subversion Repository in Blaze Advisor

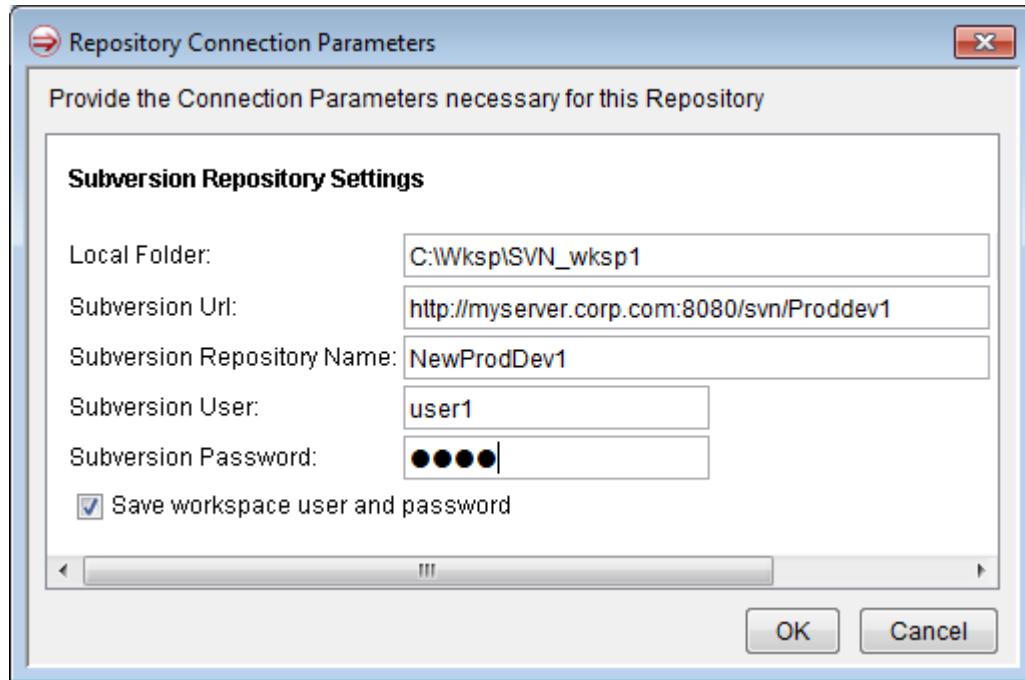
You can use the Blaze Advisor IDE or the **NdRomAdminUtil** utility to create an file repository connecting to a Subversion versioning service. In order for a repository to be successfully created, you need to ensure that Subversion is installed on a machine where you have appropriate access permissions to the parent directory location where you want to create your repository.

You can create a file SVN repository on the trunk or a branch. You enter the trunk or branch location as a URL in the Subversion Url field of the New Repositorywizard. You can use any name for the local folder including the same name as the Subversion Repository Name field but it not a requirement. After you have created a Subversion repository, you can connect to it by importing a project into the Eclipse workspace. You can verify the versioning commands in Blaze Advisor, see "["Verifying Blaze Advisor Versioning Commands" on page 230](#).



Important To avoid any creation errors, when entering values into the fields of the file Subversion Repository Settings page, leave no trailing spaces.

-  **Note** If you want to add a comment when you create an SVN Repository, you need to use the NdRomAdminUtil utility. See ["Creating a Subversion Repository Using the Utility" on page 258](#).



To create a Subversion repository

- 1 Select **Repository > New Repository** command.
- 2 Select a **File Repository (Subversion versioning)** for the **Repository Type**.
- 3 Enter a name in the **Repository Nickname** field.

When you finish creating the repository, the name you provided in this field is displayed in the list of available connections in the wizard. See ["Connecting to a Blaze Advisor Workspace or Repository" on page 52](#).

It also is added to <Name> element in the `com.blazesoftware.repository_config.cfg` file in the root directory accessible through Windows Explorer.

- 4 Enter the following information to connect to the server host machine and create the repository:

Repository Type Connection Parameter	Description
Local Folder	<p>The path to a local directory that is populated with a copy of the files from the repository. You can see these files when you connect to the repository.</p> <p>You can use any appropriate folder name for the directory. The path or directory name <i>does not</i> need to match the values in the Subversion Url or the Subversion Repository Name fields in this wizard.</p> <p>If you connect to a repository with a local or private workspace, you may need to click the Update icon to populate the workspace with the latest files from the repository before you can begin checking out files</p>
Subversion Url	<p>The URL for the machine running Subversion and the path to the directory where you have read/write access to create a repository.</p> <p>Example: <code>svn://<serverhostname>/<path to the parent directory></code></p> <p>This path can be either to the trunk or a branch. See “Creating a Subversion Repository Using the Trunk or a Branch” on page 258.</p> <p>Creation of branches and merging is not supported.</p>
Subversion Repository Name	<p>The name of the repository you want to create. The name you specify here is appended to the value you entered in the Subversion Url for the full repository path.</p>
Subversion User	User name for the server connection.
Subversion Password	Password for the server connection.

- 5 (Optional) By default the **Save workspace user and password** option is selected. If you select this option, the credentials are saved in the connection instance file in the Admin Repository but not in the repository configuration file.

If you do not want your user name and password to be persisted, clear this check box and you will need to enter your user name and password each time you want to connect to the repository. See “[Saving User Name and Passwords](#)” on page 43.

- 6 Click **OK**.

The repository creation operation may take longer than usual. When the repository is created you see a success message.

You can connect to the workspace using the Blaze Advisor IDE. See “[Connecting to a Blaze Advisor Workspace or Repository](#)” on page 52.

After you create the repository, proceed with one or more of the following tasks:

- If you need to create additional workspaces or connections to another existing workspace, see "[Creating Workspaces or Workspace Connections for a Subversion Repository](#)" on page 263.
- After you have created the repository with a local workspace, you may want to verify the versioning commands, see "[Verifying Blaze Advisor Versioning Commands](#)" on page 230.

Creating a Subversion Repository Using the Trunk or a Branch

You can create a File SVN Repository on the SVN trunk or branch. If you want to create a File SVN repository on a branch, you must have access to the branch prior to using the Blaze Advisor IDE or the `NdRomAdminUtil` utility to create the repository. For more information on creating and working with branches, see your Subversion documentation, see <http://svnbook.red-bean.com/en/1.0/ch04s02.html>.

When you are ready to create the repository using the Blaze Advisor IDE, you need to enter the name of URL for the trunk or the branch location in the Subversion Url field of the New Repository wizard in Blaze Advisor. See "[Creating a Subversion Repository in Blaze Advisor](#)" on page 255.

If you want to create the repository using the `NdRomAdminUtil` utility, see "[Creating a Subversion Repository Using the Utility](#)" on page 258.

Creating a Subversion Repository Using the Utility

You can create a File Repository connected to a Subversion open source version control system using the `NdRomAdminUtil` utility.

Before you can run the utility to create the repository, you need to write three connection configuration files and a configuration file:

- "[About Writing Files for Creating a Blaze Advisor Repository or Workspace Using the Utility](#)" on page 168
- "[Writing the Connection Configuration File for an Admin Repository](#)" on page 33
- "[Writing a Connection Configuration File for a Subversion Repository](#)" on page 258
- "[Writing a Repository Configuration File for a Subversion Repository](#)" on page 260
- "[Writing a Batch File for Creating a Subversion Repository](#)" on page 262

Writing a Connection Configuration File for a Subversion Repository

You can write a connection configuration file (.cfg) for a File repository (Subversion versioning). This connection file needs to contain the structure and the values described in the table below.

You use a local workspace with this repository type. When you create Subversion repository using the utility, you also need to specify the path to the workspace connection configuration file. See ["File Workspace" on page 186](#).



Note For reasons of brevity, a pair of element tags will be represented by the following notation, <example/>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<VersioningRepository Connection/>	Used when you need to create a connection for a Subversion or ClearCase repository. All other tags in this table are nested within this tag.
<Factory/>	Used when you want to create an instance of a class. com.blazesoft.repository.scm.NdScmServer Connection
<PersistCredentials/>	<p>(Optional) Used to save the username and password to the connection instance file in the Admin Repository.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ■ false -- do not store connection credentials in the connection instance. If you choose not to store the connection information, each time you connect to the repository in the Blaze Advisor IDE, you need to manually enter your user name and password. ■ true -- store the connection credentials. <p>See "Saving User Names and Passwords Using the Utility" on page 183.</p> <p>Note When you create an SVN or ClearCase repository using the IDE, the value for this element is always false in the generated repository configuration file. But if you selected the option to save the workspace user and password, the connection instance file persists the credentials.</p>
<RepositoryVersionSystem AdminFactory/>	Used to create an instance of the system version administrator class.
<JavaName/>	Used to specify the system version administrator class. This class is installed with Blaze Advisor. com.blazesoft.repository.scm.subversion.NdSvn VersionSystemAdmin
<VersioningSystemVersion ManagerFactory/>	Used to create an instance of the system version manager class.
<JavaName/>	Used to specify the default system version manager class. This class installed with Blaze Advisor. com.blazesoft.repository.scm.subversion.NdSvnVersionSystemVersionManager
<VersionManagerClass/>	Used to specify the version manager used. This class is installed with Blaze Advisor. com.blazesoft.repository.scm.subversion.NdSvn WorkspaceVersionManager

Tag	Description
<RepositoryName/>	Used to specify the name of the repository you want to create.
<ServerUrl/>	Used to specify the URL of the host machine or file location for the trunk or branch. Creation of branches and merging is not supported.
<User/>	Used to specify the user name for the connection to the Subversion server.
<Password/>	Used to specify the password for the connection to the Subversion server.

 **Note** To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
...
<VersioningRepositoryConnection>
<Factory> com.blazesoftware.repository.scm.NdScmServerConnection </Factory>
<RepositoryVersionSystemAdminFactory>
    <JavaName>com.blazesoftware.repository.scm.subversion.
        NdSvnVersionSystemAdmin</JavaName>
    </RepositoryVersionSystemAdminFactory>
    <VersioningSystemVersionManagerFactory>
        <JavaName>

            com.blazesoftware.repository.scm.subversion.NdSvnVersionSystemVersionManager
                <JavaName>
                </VersioningSystemVersionManagerFactory>
                <VersionManagerClass>com.blazesoftware.repository.scm.subversion.NdSvnWorksp
                    ace
                    VersionManager</VersionManagerClass>
                    <User> user1 </User>
                    <Password> pswd123 </Password>
                    <RepositoryName> repoTest1 </RepositoryName>
                    <ServerUrl> http://servername/DevTeam/RuleProj </ServerUrl>
    </VersioningRepositoryConnection>
    ...

```

Writing a Repository Configuration File for a Subversion Repository

You can use this repository configuration file when creating a File repository (Subversion versioning).

 **Note** To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```

<RomConfig>
    <RomQueryManagerConfig>
        <RomQueryManagerFactory>
            <JavaName> com.blazesoft.template.repository.query.
                NdRomDefaultQueryManager </JavaName>
            </RomQueryManagerFactory>
        </RomQueryManagerConfig>
        <RomConnectionManagerConfig>
            <ConnectionMode> 0 </ConnectionMode>
            <RomConnectionManagerFactory>
                <JavaName> com.blazesoft.template.repository.impl.NdDefault
                    RomConnectionManager</JavaName>
                </RomConnectionManagerFactory>
            </RomConnectionManagerConfig>
            <RomSchemaManagerConfig>
                <RomSchemaManagerFactory>
                    <JavaName> com.blazesoft.template.repository.impl.NdDefault
                        RomSchemaManager </JavaName>
                    </RomSchemaManagerFactory>
                </RomSchemaManagerConfig>
                <RepositoryConfig>
                    <RepositoryVersionManagerConfig>
                        <RepositoryVersionManagerFactory>
                            <JavaName>com.blazesoft.repository.scm.subversion.NdSvnWorkspace
                                VersionManager</JavaName>
                        </RepositoryVersionManagerFactory>
                        <PrivateWorkspaceSupported> true </PrivateWorkspaceSupported>
                        <VersioningRepositoryConnection>
                            <Factory> com.blazesoft.repository.scm.NdScmServerConnection
                                </Factory>
                            <RepositoryVersionSystemAdminFactory>
                                <JavaName>com.blazesoft.repository.scm.subversion.NdSvnVersionSystemAdmin
                                    </JavaName>
                                </RepositoryVersionSystemAdminFactory>
                            </VersioningRepositoryConnection>
                            <VersionManagerClass>com.blazesoft.repository.scm.NdFileScmVersioning
                                SystemVersionManager</VersionManagerClass>
                            <User> user1 </User>
                            <Password> pswd </Password>
                            <RepositoryName> repoTest1 </RepositoryName>
                            <ServerUrl> svn://<serverhostname>/<path to the parent directory
                                either branch or trunk>
                            </VersioningRepositoryConnection>
                            <Name> Version Control Configuration for CC </Name>
                            </RepositoryVersionManagerConfig>
                            <AtticRepository> false </AtticRepository>
                        </RepositoryVersionSystemAdminFactory>
                    </RepositoryVersionManagerConfig>
                </RepositoryConfig>
                <AllowUnusedValues> true </AllowUnusedValues>
            </RomSchemaManagerConfig>
        </RomConnectionManagerConfig>
    </RomQueryManagerConfig>
</RomConfig>

```

Writing a Batch File for Creating a Subversion Repository

You run the `NdRomAdminUtil` utility twice to create a File Subversion Repository. First you run the utility to create the repository and then you run it a second time to create the workspace. Each time you run the utility, you must use the correct arguments and options for each task. See “[Creating a Blaze Advisor Workspace Using the Utility](#)” on page 228.

Command, Argument, or Option	Description
<code>createRepository</code>	Command to create a repository
<code>-repositoryConnection</code>	Argument takes the path to the location of the repository connection configuration file (.cfg). See “ Repository Connection Configuration File Examples ” on page 171. This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
<code>-workspaceConnection</code> Note Use this argument to create a workspace for a BVS Repository with private workspaces, File CVS Repository File Repository (Subversion versioning) or a File Repository (ClearCase versioning) using an IBM Rational ClearCase server connection.	Argument takes the path to the location of the workspace connection configuration file (.cfg) See “ Writing a Connection Configuration File for a Private or a Local Workspace ” on page 185. This file contains information such as the repository name, the workspace connection name, the user name and password for the connection, and the path to the workspace.
<code>-systemconnection</code>	Argument takes the path to the location of the system Folder connection configuration file (.cfg). See “ Repository Connection Configuration File Examples ” on page 171. This file contains information such as the name of the repository which in this case is Admin Repository, the location of the Admin Repository and the user name.
<code>--configuration</code>	Argument takes the path to the location of the configuration file(.cfg).See “ Writing a Repository Configuration File ” on page 191.
<code>-verbose</code>	Option prints out all details messages into the command console.
<code>-m</code>	Option to add a comment when creating a repository. Note Use double quotes around your comment text.
<code>-debug</code>	Option prints out tracing information into the command console.

Example of a batch file to create a SVN repository using pathnames:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil
createRepository
-repositoryConnection C:/Repo_Connections/ConnectionFiles/DefaultRepo.cfg
-systemConnection C:/Repo_Connections/ConnectionFiles/sysAdmin.cfg -
-configuration C:/Blaze/Advisor/lib/FileSVNRepository.cfg -verbose
```

After you create the repository, you can do one or more of the following tasks:

- After you have created the repository, you need to create a connection to it in the Blaze Advisor IDE, see "[Creating a New Blaze Advisor Workspace or Repository Connection](#)" on page 58.

After you create the repository, proceed with one or more of the following tasks:

- If you need to create additional workspaces or connections to another existing workspace, see "[Creating Workspaces or Workspace Connections for a Subversion Repository](#)" on page 263.
- After you have created the repository with a local workspace, you may want to verify the versioning commands, see "[Verifying Blaze Advisor Versioning Commands](#)" on page 230.

Creating Workspaces or Workspace Connections for a Subversion Repository

When you want your users to use to connect to a workspace using a Subversion repository, you need to complete the following tasks :

- If you want your users to connect to the Subversion repository using the Blaze Advisor IDE, you can do one of the following:
 - Create a new connection to an existing workspace
See "[Creating a New Blaze Advisor Workspace or Repository Connection](#)" on page 58.
 - Create a new workspace
See "[Creating a Local or Private Workspace](#)" on page 226.
- When you generate an RMA, you can decide whether to use the current workspace or create one specifically for use with the RMA. For more information see, "Selecting Workspace Options" in *DevelopingRuleMaintenanceApplications.pdf*.

Implementing a ClearCase Repository

You can create a ClearCase repository connecting to an IBM Rational ClearCase version 8.0.1 server using the Blaze Advisor IDE or the `NdRomAdminUtil` utility.

This section contains the following topics:

- "[Prerequisites for Creating a ClearCase Repository](#)" on page 264
- "[Creating a ClearCase Repository in the Blaze Advisor IDE](#)" on page 267
- "[Creating a ClearCase Repository Using the Utility](#)" on page 270
- "[Creating ClearCase Workspace Connections](#)" on page 275
- "[Creating RMA ClearCase Workspace Connections](#)" on page 277

If you plan to check in items outside the Blaze Advisor context, see "[Using the External SCM Versioning System Outside the Blaze Advisor Context](#)" on page 284.

**Note**

- Due to limitations in the IBM ClearCase API, the Locked By Other User overlay icon does not display in the IDE when a user attempts to check out an entity that is already checked out by another user. In the RMA, the Locked By Other User icon does not display in the Status column in the Project Explorer.
- We support one user per single session of the Blaze Advisor IDE. To login as another user, the Blaze Advisor IDE has to be restarted or a new instance of Eclipse needs to be opened for the new user.

Prerequisites for Creating a ClearCase Repository

Before you can create a ClearCase repository, you need to ensure that the source control management system you want to use is installed on a network machine and that you have appropriate access permissions.

There are several prerequisites for creating a file ClearCase repository:

- ["IBM Rational ClearCase Jar Files" on page 264](#)
- ["Creating a VOB" on page 265](#)
- ["Creating a View on the IBM ClearCase Server" on page 265](#)
- ["IBM Rational ClearCase Views and Blaze Advisor Workspaces" on page 267](#)



Note The IBM Rational® ClearTeam Explorer™ 8.0.1 is supported for creating views, which are the equivalent of Blaze Advisor workspaces.

IBM Rational ClearCase Jar Files

Prior to creating a File repository (ClearCase versioning) the Blaze Advisor IDE in Eclipse, you need to add several ClearCase jar files from your ClearCase installation to the ClearCase jar files that are available with your Blaze Advisor installation to the eclipse dropins directory.

To add ClearCase jar files

- 1 Copy the following jar files from your ClearCase server installation:
 - <CLEARCASE_INSTALL_DIR>/common/stpwcm.jar
 - <CLEARCASE_INSTALL_DIR>/common/stpcmmn.jar
 - <CLEARCASE_INSTALL_DIR>/clearcase/web/teamapi/remote_core.jar
 - <CLEARCASE_INSTALL_DIR>/clearcase/web/teamapi/stpcc.jar
- 2 Paste these jar file to: <ADVISOR_HOME>/EclipseDropins/com.fairisaac.eclipse.clearcasewrapper in your Blaze Advisor installation. In this directory, the following jar files are already available:
 - commons-codec-1.3.jar
 - commons-httpclient-3.0.jar
 - commons-logging-api-1.1.1.jar

- 3 Copy the `com.fairisaac.eclipse.clearcasewrapper` directory to the `<ECLIPSE_HOME>/dropins` directory.
The `<ECLIPSE_HOME>` directory is the location of your Eclipse installation, for example if you installed Eclipse to your `C:/` directory your `<ECLIPSE_HOME>` is `C:/eclipse`.
The `MANIFEST.MF` located in the `<ADVISOR_HOME>/EclipseDropins/com.fairisaac.eclipse.clearcasewrapper/META-INF` already lists the ClearCase jar files you added.
- 4 After you have completed the other prerequisites, you can launch Eclipse.

If you are deploying an RMA from a project created in a ClearCase repository, you need to ensure that the ClearCase jar files are added to the `WEB-INF/lib` directory. See "Adding Jar Files for Your Repository Type" in *DevelopingRuleMaintenanceApplications.pdf*.

Creating a VOB

A *versioned object* (VOB) is an object in the file ClearCase repository. It is the root of the hierarchy of versioned elements. A VOB can contain versioned *elements* of different types. An element is the equivalent of a Blaze Advisor file. If you already have a VOB, you can use it to create a file ClearCase repository. If you do not have a VOB, you need to create a storage VOB and register it in the VOB registry files on the network, prior to creating a ClearCase repository. You need to have ClearCase administrator privileges to create a VOB.

Creating a VOB on a UNIX Server

To create a *public* VOB, you use the IBM ClearCase command-line tool, `cleartool`. A public VOB can be accessed by all users who need to connect to the ClearCase repository in Blaze Advisor. Each user connecting to the same ClearCase repository needs to have their own view loaded with the same VOB. See "["Creating a View on the IBM ClearCase Server" on page 265](#)".

Creating a View on the IBM ClearCase Server

After you create a VOB, you create a view using the IBM Rational® ClearTeam Explorer™ version 8.0.1. A view is similar to a Blaze Advisor workspace. You load a copy of a VOB into your view. See "["IBM Rational ClearCase Views and Blaze Advisor Workspaces" on page 267](#)".

The ClearTeam Explorer is an standalone Eclipse-based application that allows you to perform most ClearCase operations including the creation of views. For more information about the ClearTeam Explorer, see:

http://www-01.ibm.com/support/knowledgecenter/SSSH27_8.0.1/com.ibm.rational.clearcase.help.ic.doc/helpindex_ccrc.htm

You must be able to login to the IBM ClearCase server using the ClearTeam Explorer. If you are not able to successfully login, contact your system administrator.

To create a view using the IBM Rational ClearTeam Explorer

- 1 Create a subdirectory on your local machine to store the ClearCase views.
- 2 Launch ClearTeam Explorer.
- 3 If this is the first time you are attempting to create a 8.0.1 view, then you need to click **ClearCase > Connect** and enter the following information:
 - a **Server URL:** <serverhost machine>.
 - b **User name:** <your username to access serverhost machine>.
 - c **Password:** <your password to access serverhost machine>.
 - d (Optional) Select the **Store and reuse credentials** option.
 - e Click **OK**.
- 4 Select **ClearCase > Create View** and select a server for your new view.
- 5 Use the **Create a base ClearCase view** option selected by default and click **Next**.
- 6 Retain the View Type as Web and for the **View-Tag** field, ensure that your user name is included in the name of the view.

If you plan to have users connect to the repository through an RMA, you need to ensure that the required directory structure for the local folder is used. See ["Generating an RMA with a ClearCase Workspaces Directory Parameter" on page 277](#).
- 7 Click **View Options** and ensure that the Interop Text Mode (for mixed UNIX/Windows environments) is set to **Transparent(do not alter file content)**.
- 8 Click **OK**.
- 9 For the **Copy area pathname** field, enter the path to the directory you created to store the ClearCase views.
- 10 Click **Finish**.
- 11 On the **Edit Configuration** dialog, you can select a VOB and enter or edit options for the **Load Rules, Version Selection Rules, or Copy Rules**.
 - On the **Load Rules** tab you can select the resources to display and which set of files and directories are loaded into your view.

You can expand the VOBs directory to select the VOB you want to use.
 - On the **Version Selection Rules** tab you can select which version of each of those files and directories is chosen.
 - On the **Copy Rules** tab you can select a view or views where you want to copy rules.
- If you are creating a view for each member of your team, see ["Creating a View For Each User" on page 267](#).
- 12 Click **Finish**.

Creating a View For Each User

If you are an administrator and you want to provide access to the ClearCase repository for other users, you need to return to the ClearTeam Explorer to create a view for each user. The steps you take, depend on which interface your users need to use when connecting to the repository:

- If you want your users to use the Blaze Advisor IDE when connecting to the repository, after creating a new view for each user and populating the repository folder from the VOB to each new view, you need to edit the repository configuration file with the path to each user's workspace before they can create a repository connection. See "[Editing the Repository Configuration File](#)" on page 276.
- If you want your users to use an RMA when connecting to the repository, prior to creating the view, you need to ensure that each user uses same directory structure for their local folder. In addition, when creating their view, the view name needs to include their ClearCase username. See "[Generating an RMA with a ClearCase Workspaces Directory Parameter](#)" on page 277.

IBM Rational ClearCase Views and Blaze Advisor Workspaces

An IBM Rational ClearCase view is similar in concept to a Blaze Advisor workspace. The view is managed through a *viewstore*. The *viewstore* is a directory where one or more views may be created and stored. You can create a view through the IBM Rational ClearTeam Explorer. In general, after you create the ClearCase repository, your users can update the workspace files in their views through Blaze Advisor.

A Blaze Advisor workspace is a physical location on a local machine that is connected to the repository on the server. As an administrator, when you create a File repository (ClearCase versioning), you specify a view location (Local Folder) on your machine. See "[Creating a ClearCase Repository in the Blaze Advisor IDE](#)" on page 267.

Each user needs a workspace to connect to the ClearCase through the Blaze Advisor IDE or an RMA, see "[Creating ClearCase Workspace Connections](#)" on page 275.

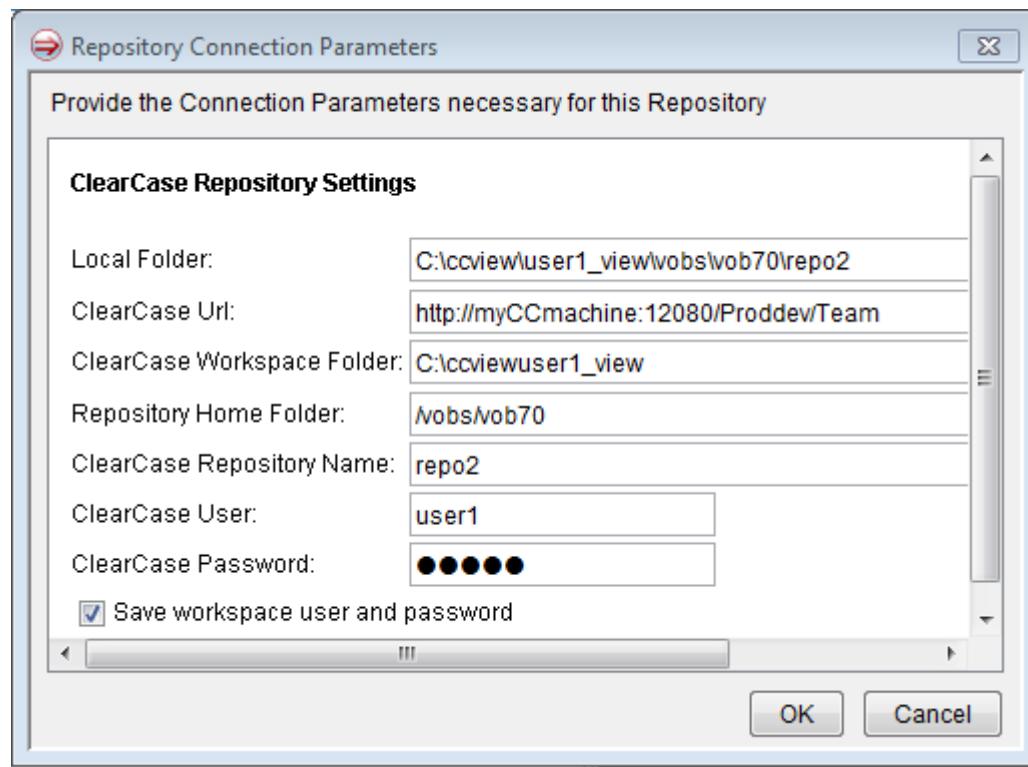
Creating a ClearCase Repository in the Blaze Advisor IDE

If you have created a VOB and loaded it into a view, you can create a ClearCase repository using the Blaze Advisor IDE. Below is an example of the values you need to enter for the connection parameters.

After you have created a ClearCase repository, you can verify the versioning commands in the Blaze Advisor IDE, see "[Verifying Blaze Advisor Versioning Commands](#)" on page 230.



Important To avoid any creation errors, when entering values into the fields of the ClearCase Repository Settings page, leave no trailing spaces.



To create a ClearCase repository

- 1 Select **Repository > New Repository** command.
- 2 Select a **File Repository (ClearCase versioning)** for the **Repository Type**.
- 3 Enter a name for the repository connection in the Repository Nickname field.
The name you provide in this field is displayed in the list of available connections in the wizard in Blaze Advisor. See "[Connecting to a Blaze Advisor Workspace or Repository](#)" on page 52.
- 4 Enter the following information to connect to the machine that is hosting the server and create the repository:

Repository Type Connection Parameter	Description
Local Folder	<p>The full path to the workspace on the local machine.</p> <p>The location becomes:</p> <p><ClearCase Workspace Folder>/ <Repository Home Folder>/ <ClearCaseRepositoryName></p> <p>Example: C:\ccview\vos\myVob\repo1</p>
ClearCase Url	<p>The URL for the location on the serverhost machine.</p> <p>Example: http://<machineName>:<port number>/<directorylocation></p> <p>If you want to create a repository on a branch, see “Creating a ClearCase Repository Using a Branch” on page 270.</p>
ClearCase Workspace Folder	<p>The path to your ClearCase view directory.</p> <p>Example: C:/ccview</p>
Repository Home Folder	<p>The folder inside the view where the VOB is located. This is considered to be the path for repository root.</p> <p>Example: /vos/myVob</p>
ClearCase Repository Name	<p>Name of the repository you want to create in the VOB.</p> <p>Example: repo1</p>
ClearCase User	User name for the server connection.
ClearCase Password	Password for the server connection.

- 5 (Optional) By default the **Save workspace user and password** option is selected. If you select this option, the credentials are saved in the connection instance file in the Admin Repository but not in the repository configuration file.

If you do not want your user name and password to be persisted, clear this check box and you will need to enter your user name and password each time you want to connect to the repository. See [“Saving User Name and Passwords” on page 43](#).

- 6 Click **OK**.

The repository creation operation may take longer than usual. When the repository is created you see a success message.

After you exit the wizard, you can connect to the workspace using the Blaze Advisor IDE. See [“Connecting to a Blaze Advisor Workspace or Repository” on page 52](#).

After you create the repository, proceed with one or more of the following tasks:

- If you want to create a new workspace connection, see [“Creating ClearCase Workspace Connections” on page 275](#).

- After you have created the repository with a local workspace, you may want to verify the versioning commands, see "[Verifying Blaze Advisor Versioning Commands](#)" on page 230.
- If you are using a File ClearCase repository and want to change the history view displayed in the History tab, see "[Changing the History View](#)" on page 279.

Creating a ClearCase Repository Using a Branch

If you want to use a branch as the location for your ClearCase repository, you need to create a ClearCase branch prior to using the Blaze Advisor IDE or the `NdRomAdminUtil` utility to create the repository. For more information on creating and working with branches, see your IBM Rational ClearCase documentation.

When you have created your ClearCase branch, you need to create a new repository based on that branch using the URL for the branch in the ClearCase Url field. See "[Creating a ClearCase Repository in the Blaze Advisor IDE](#)" on page 267 or "[Creating a ClearCase Repository Using the Utility](#)" on page 270.

Creating a ClearCase Repository Using the Utility

You can create a File Repository (ClearCase versioning) connected to a IBM Rational ClearCase server by using the `NdRomAdminUtil` utility. Before you can run the utility to create the repository, you need to write three connection configuration files and a configuration file.

- "[About Writing Files for Creating a Blaze Advisor Repository or Workspace Using the Utility](#)" on page 168
- "[Writing the Connection Configuration File for an Admin Repository](#)" on page 33
- "[Writing the Connection Configuration File for a ClearCase Repository](#)" on page 270
- "[Writing the Repository Configuration File for ClearCase Repository](#)" on page 273
- "[Writing a Batch File for Creating a ClearCase Repository](#)" on page 274

After you have created the repository, you can connect to it in Blaze Advisor. See "[Creating ClearCase Workspace Connections](#)" on page 275.

After creating the repository if you want to verify the versioning commands, see "[Verifying Blaze Advisor Versioning Commands](#)" on page 230.

Writing the Connection Configuration File for a ClearCase Repository

You can write a connection configuration file (.cfg) for a File repository (ClearCase versioning). This connection file needs to contain the structure and the values described in the table below.

You use a local workspace with this repository type. When you create ClearCase repository using the utility, you also need to specify the path to the workspace connection configuration file. See "[File Workspace](#)" on page 186.



Note For reasons of brevity, a pair of element tags will be represented by the following notation, <example/>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<VersioningRepository Connection/>	Used when you need to create a connection for a Subversion or ClearCase repository. All other tags in this table are nested within this tag.
<Factory/>	Used when you want to create an instance of a class. com.blazesoft.repository.scm.NdScmServer Connection
<PersistCredentials/>	<p>(Optional) Used to save the username and password to the connection instance file in the Admin Repository.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ■ false -- do not store connection credentials in the connection instance. If you choose not to store the connection information, each time you connect to the repository in the Blaze Advisor IDE, you need to manually enter your user name and password. ■ true -- store the connection credentials. <p>See "Saving User Names and Passwords Using the Utility" on page 183.</p> <p>Note When you create an SVN or ClearCase repository using the IDE, the value for this element is always false in the generated repository configuration file. But if you selected the option to save the workspace user and password, the connection instance file persists the credentials.</p>
<RepositoryVersionSystem AdminFactory/>	Used to create an instance of the system version administrator class.
<JavaName/>	Used to specify the system version administrator class. This class is installed with Blaze Advisor. com.blazesoft.repository.scm.wvcm.NdFileWvcm VersionSystemAdmin
<VersioningSystemVersion ManagerFactory/>	Used to create an instance of the system version manager class.
<JavaName/>	Used to specify the default system version manager class. This class installed with Blaze Advisor. com.blazesoft.repository.scm.wvcm.cc.NdCcVersioningSystemVersionManager
<VersionManagerClass/>	Used to specify the version manager used. This class is installed with Blaze Advisor. com.blazesoft.repository.scm.wvcm.cc.NdCcVersion Manager

Tag	Description
<ScmProvider/>	Used to specify the class used for the ClearCase repository type. Example: com.ibm.rational.stp.client.internal.cc.CcOnlyProviderImpl
<WorkspaceFolder/>	Used to specify the root of the view.
<RepositoryHomeFolder/>	Used to specify the name of the VOB you want to use. Example: /vobs/yourVOB
<RepositoryName/>	Used to specify the name of the repository you want to create.
<ServerUrl/>	Used to specify the URL of the host machine with the Web client.
<ScmBranch>	If you already have a branch, you can specify its location using this tag. Creation of branches and merging is not supported.
<User/>	Used to specify the user name for the connection to the IBM Rational ClearCase server.
<Password/>	Used to specify the password for the connection to the IBM Rational ClearCase server.

 **Note** To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```

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

<VersioningRepositoryConnection>
<Factory> com.blazesoftware.repository.scm.NdScmServerConnection </Factory>
    <RepositoryVersionSystemAdminFactory>

        <JavaName>com.blazesoftware.repository.scm.wvcm.NdFileWvcmVersionSystemAdmin
            </JavaName>
        </RepositoryVersionSystemAdminFactory>
        <VersioningSystemVersionManagerFactory>

            <JavaName>com.blazesoftware.repository.scm.wvcm.cc.NdCcVersioningSystemVersion
                Manager</JavaName>
            </VersioningSystemVersionManagerFactory>
            <VersionManagerClass>com.blazesoftware.repository.scm.wvcm.cc.NdCcVersionManager
            </VersionManagerClass>

<ScmProvider>com.ibm.rational.stp.client.internal.cc.CcOnlyProviderImpl
</ScmProvider>
<WorkspaceFolder>C:\cc_view</rootOfView>
<RepositoryHomeFolder>/vobs/myvob</RepositoryHomeFolder>
<User> user1 </User>

```

```

<Password> pswd123 </Password>
<RepositoryName> repoTest1 </RepositoryName>
<ServerUrl> http://serverHost01/ccrc </ServerUrl>
</VersioningRepositoryConnection>

```

Writing the Repository Configuration File for ClearCase Repository

You can use this repository configuration file when creating a File repository (ClearCase versioning) connected to an IBM Rational ClearCase server.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```

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

<RomConfig>
    <RomQueryManagerConfig>
        <RomQueryManagerFactory>
            <JavaName> com.blazesoft.template.repository.query.
                NdRomDefaultQueryManager </JavaName>
        </RomQueryManagerFactory>
    </RomQueryManagerConfig>
    <RomConnectionManagerConfig>
        <ConnectionMode> 0 </ConnectionMode>
        <RomConnectionManagerFactory>
            <JavaName> com.blazesoft.template.repository.impl.NdDefault
                RomConnectionManager</JavaName>
        </RomConnectionManagerFactory>
    </RomConnectionManagerConfig>
    <RomSchemaManagerConfig>
        <RomSchemaManagerFactory>
            <JavaName> com.blazesoft.template.repository.impl.NdDefault
                RomSchemaManager </JavaName>
        </RomSchemaManagerFactory>
    </RomSchemaManagerConfig>
    <RepositoryConfig>
        <RepositoryVersionManagerConfig>
            <RepositoryVersionManagerFactory>
                <JavaName> com.blazesoft.repository.scm.wvcm.cc.NdCcVersion
                    Manager </JavaName>
            </RepositoryVersionManagerFactory>
            <PrivateWorkspaceSupported> true </PrivateWorkspaceSupported>
            <VersioningRepositoryConnection>
                <Factory> com.blazesoft.repository.scm.NdScmServerConnection </
                    Factory>
                <RepositoryVersionSystemAdminFactory>
                    <JavaName> com.blazesoft.repository.scm.wvcm.NdFileWvcmVersion
                        SystemAdmin </JavaName>
                </RepositoryVersionSystemAdminFactory>
                <VersioningSystemVersionManagerFactory>
                    <JavaName> com.blazesoft.repository.scm.wvcm.cc.NdCcVersioning
                        SystemVersionManager </JavaName>
                </VersioningSystemVersionManagerFactory>
            </VersioningRepositoryConnection>
        </RepositoryVersionManagerConfig>
    </RepositoryConfig>
</RomConfig>

```

```
<VersionManagerClass>com.blazesoftware.repository.scm.wvcm.cc.NdCcVersion
    Manager</VersionManagerClass>

<ScmProvider>com.ibm.rational.stp.client.internal.cc.CcOnlyProviderImpl
    </ScmProvider>
    <WorkspaceFolder>C:\ccview</WorkspaceFolder>
    <RepositoryHomeFolder>/vobs/myvob</RepositoryHomeFolder>
    <User> user1 </User>
    <Password> pswd </Password>
    <RepositoryName> repoTest1 </RepositoryName>
    <ServerUrl> http://serverHost/ccrc </ServerUrl>
    </VersioningRepositoryConnection>
    <Name> Version Control Configuration for CC </Name>
    </RepositoryVersionManagerConfig>
    <AtticRepository> false </AtticRepository>
</RepositoryConfig>

<AllowUnusedValues> true </AllowUnusedValues>

</RomConfig>
```

Writing a Batch File for Creating a ClearCase Repository

You run the `NdRomAdminUtil` utility twice to create a File ClearCase Repository. First you run the utility to create the repository and then you run it a second time to create the workspace. Each time you run the utility, you must use the correct arguments and options for each task. See [“Creating a New Blaze Advisor Workspace or Repository Connection” on page 58](#).

Command, Argument, or Option	Description
<code>createRepository</code>	Command to create a repository
<code>-repositoryConnection</code>	Argument takes the path to the location of the repository connection configuration file (.cfg). See “Repository Connection Configuration File Examples” on page 171 . This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
<code>-workspaceConnection</code> Note Use this argument to create a workspace for a BVS Repository with private workspaces, File CVS Repository File Repository (Subversion versioning) or a File Repository (ClearCase versioning) using an IBM Rational ClearCase server connection.	Argument takes the path to the location of the workspace connection configuration file (.cfg) See “Writing a Connection Configuration File for a Private or a Local Workspace” on page 185 . This file contains information such as the repository name, the workspace connection name, the user name and password for the connection, and the path to the workspace.

Command, Argument, or Option	Description
-systemconnection	Argument takes the path to the location of the system Folder connection configuration file (.cfg). See "Repository Connection Configuration File Examples" on page 171 . This file contains information such as the name of the repository which in this case is Admin Repository, the location of the Admin Repository and the user name.
--configuration	Argument takes the path to the location of the configuration file(.cfg).See "Writing a Repository Configuration File" on page 191 .
-verbose	Option prints out all details messages into the command console.
-m	Option to add a comment when creating a repository. Note Use double quotes around your comment text.
-debug	Option prints out tracing information into the command console.

Example of a batch file to create a Clearcase repository using pathnames:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil
createRepository
-repositoryConnection C:/Repo_Connections/ConnectionFiles/DefaultRepo.cfg
-systemConnection C:/Repo_Connections/ConnectionFiles/sysAdmin.cfg -
-configuration C:/Blaze/Advisor/lib/FileClearCaseRepository.cfg -verbose
```

After you create your repository, you can do one or more of the following:

- Create a workspace connection for the repository you just created in the Blaze Advisor IDE, see ["Creating ClearCase Workspace Connections" on page 275](#).
You can also use the information in this topic if you need to create a new connection to another existing workspace or a new workspace.
- After you have created the repository with a local workspace, you may want to verify the versioning commands, see ["Verifying Blaze Advisor Versioning Commands" on page 230](#).

Creating ClearCase Workspace Connections

When you want your IDE and RMA users to connect to a workspace using a ClearCase repository, you need to complete the following tasks. The tasks are the same whether the repository is located on a ClearCase branch or the main trunk.

- Create a ClearCase view for each user
See ["Creating a View For Each User" on page 267](#).
- Edit the repository configuration file in the loaded VOB so that they can connect to the ClearCase repository.
See ["Editing the Repository Configuration File" on page 276](#).

- Create a new connection to an existing workspace
See “[Creating a New Blaze Advisor Workspace or Repository Connection](#)” on page 58.

 **Note** Creation of new ClearCase workspaces from client applications like Blaze Advisor is not supported by ClearCase. You must use ClearCase to create a workspace to use with Blaze Advisor.

If you want your users to connect to the ClearCase repository using the RMA, you need to create connections to RMA workspaces for each of your users, see “[Creating RMA ClearCase Workspace Connections](#)” on page 277.

Editing the Repository Configuration File

Initially, when you create a ClearCase repository, the repository configuration file located in the repository directory contains your specific connection credentials and workspace location. This configuration file is used to identify your user credentials and cannot be used by others.

If you want multiple users to have access to the ClearCase repository in Blaze Advisor, you need to follow these steps:

To edit the repository configuration file

- 1 Create a view for each user, see “[Creating a View on the IBM ClearCase Server](#)” on page 265.
- 2 Locate the `com_blaresoft_repository_config.cfg` in the repository directory of the VOB you loaded into the view.
- 3 Right-click on the `com_blaresoft_repository_config.cfg`, choose Properties and clear the Read-only check box to make this file editable.
- 4 Open the `com_blaresoft_repository_config.cfg` located in the repository directory in a text editor.
 - a If the credentials have been persisted:
 - Locate the `<User>` tag and replace the existing value with your user name.
 - Locate the `<Password>` tag and replace the existing value with your password.
 - b Locate the `<WorkspaceFolder>` tag and replace the existing path with the absolute path to the root of the user’s ClearCase view.
- 5 Save your changes.
- 6 Repeat Step 3 to Step 5 for each user.

Each user can now create a new connection to the repository using their view. See “[Connecting to a Blaze Advisor Repository From a Rule Maintenance Application](#)” on page 55.

If you want your users to connect to your ClearCase repository through an RMA, see “[Creating RMA ClearCase Workspace Connections](#)” on page 277.

Creating RMA ClearCase Workspace Connections

If you want your users to connect to an ClearCase Workspace when they logon to an RMA and you have not created a ClearCase workspace for them, see “[Creating ClearCase Workspace Connections](#)” on page 275.:

If you have already created a ClearCase workspace for your users, there are some additional steps you need to complete:

- “[Generating an RMA with a ClearCase Workspaces Directory Parameter](#)” on page 277
- “[Creating an RMA Workspace Connection File](#)” on page 278

Generating an RMA with a ClearCase Workspaces Directory Parameter

You can generate an RMA with a Workspaces directory parameter to provide your users with the ability to access the RMA and connect to their ClearCase workspace. To use this method, you need to ensure that each user’s local folder has the required directory structure:

<Location of the ClearCase workspace folder (view root)>\<Location of the VOB root>

In your RMAD, you need to select the **Use the absolute path to the current workspace location** option as well as the **Connect to current workspace for this user** option in the **Workspaces** tab. This ensures that the workspace information is easily editable. See “Selecting Workspace Options” in *DevelopingRuleMaintenanceApplications.pdf*.

When you or your users login into an RMA connected to a ClearCase repository, you will connect to the ClearCase view workspace.

To create an RMA workspace connection to a ClearCase Repository

- 1 Select **Project > Rule Maintenance Applications > Generate Rule Maintenance Applications**.
- 2 Select an existing or create a new RMA definition file where both the following options are selected in the Workspaces tab:
 - **Use an absolute path for the current workspace location.**
 - **Connect to current workspace for this user.**
- 3 Use the default RMA Deployment.
- 4 Specify or browse to an **RMA directory**.

- 5 Enter the following path with a parameter in the **Workspaces directory**:
`<Location of the ClearCase workspace folder>\%username%\<Location of the VOB root>`

Example: C:\ccview\%username%\vobs\myvob

This directory structure is a requirement when you specify a path with the %username% parameter because at runtime, the parameter is replaced with the actual user name entered on the RMA sign-in page.

The username in this context is also used for the RMA logon and to connect to the ClearCase view/workspace.



Note The workspaces folder that is generated for an RMA connected to a BVS or CVS repository is not supported for ClearCase. In some cases, you may see a workspaces folder created but it will be empty. This is a known issue.

- 6 Ensure that you create a connection file for each user who will be logging onto the RMA. See "[Creating an RMA Workspace Connection File](#)" on page 278.

Creating an RMA Workspace Connection File

When you want your users to log onto your RMA and you are using a ClearCase repository to track your instance file changes, you need to create an RMA connection file for each of your users.

Initially, when you generate an RMA with the ClearCase Workspaces Directory Parameter, and you log onto the RMA, a connection file is created for you. You can copy and edit the connection file for each user that you want to have access to the RMA.

For more information on generating an RMA connected to a ClearCase repository that allows access for multiple users, see "[Generating an RMA with a ClearCase Workspaces Directory Parameter](#)" on page 277.

To edit the connection file for each user

- 1 Locate the connection file that was generated for you, <yourUsername>.connection.
The connection file is located at <RMA Directory>/rma/connections directory.
- 2 Copy the connection file and save it using the following format,
`<username>.connection` where the username is the one that belongs to the team member who will be logging onto the RMA and connecting to their ClearCase workspace. Repeat this step for each team member who needs access to the same RMA.
- 3 Edit the contents of the connection file so that each team member that you want to have access to the RMA can successfully logon:
You need to search for the following parameters and edit their contents with the user-specific values:

Parameter	Value
@cm	Lines with this prefix are used for the initial set up of the repository connection and do not need to be edited in the subsequent connection files you create for each team member.
com.blazesoftware.repository.repositoryfolder	Edit with the name of the user's view. Example: C:\\\\ClearCaseViews\\\\user1_view_100\\\\admin_Test_VOB_2\\\\ccRep51
com.blazesoftware.repository.username	Edit with the name of the user. Example: user1

Example:

```
com.blazesoftware.repository.connectionclass=com.blazesoftware.repository.file.NdFileRepositoryConnection
 @_cm_com.blazesoftware.scm.connection.repository.home.folder=/
 admin_Test_VOB_2
 @_cm_com.blazesoftware.repository.username=user1
 com.blazesoftware.repository.encryption.cipher.parameter.factory=com.blazesoftware.util.crypto.java.NdJavaTripleDESKeySpecFactory
 @_cm_com.blazesoftware.scm.connection.provider=com.ibm.rational.stp.client.internal.cc.CcNetworkProviderImpl
 com.blazesoftware.repository.isForWorkspace=true
 @_cm_com.blazesoftware.repository.connectionclass=com.blazesoftware.repository.scm.NdScmServerConnection
 com.blazesoftware.repository.repositoryfolder=C:\\\\ClearCaseViews\\\\user1_view_100\\\\admin_Test_VOB_2\\\\ccRep51
 com.blazesoftware.repository.encryption.manager.class=com.blazesoftware.repository.base.NdDefaultRepositoryEncryptionManager
 @_cm_com.blazesoftware.scm.connection.url=http://companymachine:8080/ccrc
 @_cm_com.blazesoftware.repository.repositoryname=ccRep51
 @_cm_com.blazesoftware.scm.connection.workspace.folder=C:\\\\ClearCaseViews\\\\user1_view_100
 com.blazesoftware.repository.username=user1
 @_cm_com.blazesoftware.scm.version_system.admin.class=com.blazesoftware.repository.scm.wvcm.cc.NdCcVersionSystemAdmin
 @_cm_com.blazesoftware.scm.workspace.version.manager.class=com.blazesoftware.repository.scm.wvcm.cc.NdCcVersionManager
 @_cm_com.blazesoftware.scm.system.version.manager.class=com.blazesoftware.repository.scm.wvcm.cc.NdCcVersioningSystemVersionManager
 com.blazesoftware.repository.encryption.cipher=TripleDES
```

Changing the History View

In Blaze Advisor, the history view of an item is what you see displayed in the History tab in the Blaze Advisor IDE or the RMA. By default, what you see when you view the history

of an item is the record of all ancestral revisions of the current branch of your workspace. In ClearCase, you have the option to view the history of an item that includes the full history of all versions on all branches regardless of whether or not the item is a predecessor.

If you need to see the ClearCase non-ancestral history view of items in the Blaze Advisor IDE or the RMA, you can change the value of `nd.versionManager.ancestralHistory` system property by adding the following flag in your batch scripts or a command-line:

`-Dnd.versionManager.ancestralHistory=false`

This flag changes the contents displayed in the History tab in the Blaze Advisor IDE or the RMA to show all changes on all branches for an item. If you want to see the default Blaze Advisor history view, you can change the value of this flag to `true` or omit the flag.

Migrating an SCM Repository Using IBM Rational ClearCase 7.1

Blaze Advisor 7.3 supports ClearCase 8.0.1 and no longer supports 7.1. You need to migrate your ClearCase server and clients according to the IBM Rational ClearCase migration guide lines.

The following section describes post-ClearCase 7.1 migration steps that need to be applied before you can successfully connect to your existing repository. See ["Prerequisites For Updating a Repository Configuration" on page 280](#).

You can update your repository configuration using either one of these methods:

- ["Updating Your ClearCase Repository Configuration Using ClearTeam Explorer" on page 281](#)
- ["Updating Your ClearCase Repository Configuration Using the NdRomAdminUtil Utility" on page 281](#)



Important All workspaces must be ClearCase 8.0.1 workspaces connecting to ClearCase 8.0.1server.

Prerequisites For Updating a Repository Configuration

Prior to updating your repository configuration, you need to complete the following tasks:

- Migrate your ClearCase server and clients according to the IBM Rational ClearCase migration guidelines.
- Ensure that your CLASSPATH points to the IBM Rational ClearCase 8.0.1 jar files. See ["IBM Rational ClearCase Jar Files" on page 264](#).
- Create a web-view using IBM Rational ClearTeam Explorer version 8.0.1.

You can migrate your repository using either the ClearTeam Explorer or the NdRomAdminUtil utility, see:

- “[Updating Your ClearCase Repository Configuration Using ClearTeam Explorer](#)”
- “[Updating Your ClearCase Repository Configuration Using the NdRomAdminUtil Utility](#)” on page 281

Updating Your ClearCase Repository Configuration Using ClearTeam Explorer

You can use the ClearTeam Explorer to check out and make changes to your repository configuration when you migrate your repository from using ClearCase 7.1 to 8.0.1.

To edit your repository files

- 1 Checkout and edit the `com.blazesoftware.repository_config.cfg` AND `com.blazesoftware.repository_config.cfg.innovator.attbs` files of your Blaze Advisor repository.
- 2 Change the values of the following tags:
 - `<ServerUrl/>`
Point to the new TeamWeb/services/Team URL.
 - `<ScmProvider/>`
Enter the following class for:
`com.ibm.rational.stp.client.internal.cc.CcOnlyProviderImpl`
- 3 Check in both files.
- 4 You need to create new workspaces for your users.

Updating Your ClearCase Repository Configuration Using the NdRomAdminUtil Utility

You can use the `NdRomAdminUtil` utility to commit the changes you make to your repository configuration when you migrate your repository from using ClearCase 7.1 to 8.0.1.

To change your repository configuration files

- 1 Copy the `com.blazesoftware.repository_config.cfg` to a safe location.
- 2 Change the values of the following tags in this file:
 - `<ServerUrl/>`
Point to the new TeamWeb/services/Team URL.
 - `<ScmProvider/>`
Enter the following class for:
`com.ibm.rational.stp.client.internal.cc.CcOnlyProviderImpl`
- 3 Use the `NdRomAdminUtil` utility and the `changeConfig` command to change the configuration.
- 4 Test the repository connection.

- 5 Use the `NdRomAdminUtil` utility and the `commitConfig` command to commit the configuration to the versioning system.
- 6 You need to create new workspaces for your users.

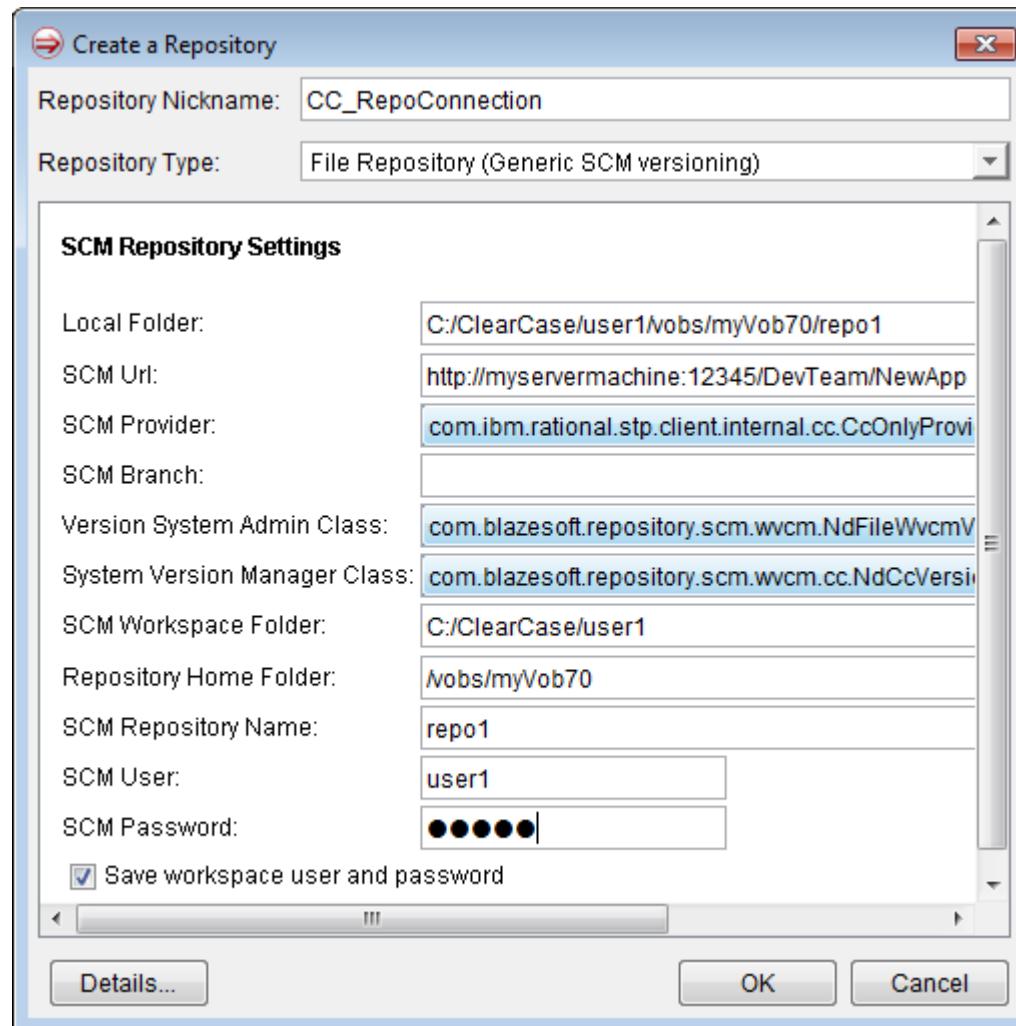
Connecting to an SCM Generic Repository

Important If you created an SCM repository connecting to an older version of IBM Rational ClearCase, this repository type is now deprecated as of Blaze Advisor 7.3.

To migrate the older repository, see “[Migrating an SCM Repository Using IBM Rational ClearCase 7.1](#)” on page 280.

To create any new ClearCase repositories, see “[Implementing a ClearCase Repository](#)” on page 263.

These instructions assume that you have already created a view and loaded the VOB containing the repository.



To create a new workspace

- 1 Select **Repository > New Repository**.
- 2 Select **File Repository (Generic SCM versioning)**.
- 3 Enter a **Repository NickName** for the repository connection.
This connection name is the same one that is in your `com.blazesoftware_repository_config.cfg` file.
- 4 Click **OK**.
- 5 Enter values for the following fields:

Repository Type Connection Parameter	Description
Local Folder	<p>The local path to the view directory. The Local Folder field is displayed if you have selected a repository type with a private or local workspace such as a File SCM repository or a BVS repository with a private workspace.</p> <p>Important When creating a File SCM ClearCase repository, the path you enter is the full path to the workspace.</p> <p>The location becomes: <code><SCM Workspace Folder>/<Repository Home Folder>/<ClearCaseRepositoryName></code></p> <p>Example: <code>C:\ccview\vos\myVob\repo1</code></p>
SCM Url	<p>The url for the IBM Rational ClearCase Web interface.</p> <p>Example: <code>http://<machineName>/ccrc</code></p>
SCM Provider	<p>The IBM Rational ClearCase provider class name.</p> <p>Example: <code>com.ibm.rational.stp.client.internal.core.ClientHostedProvider</code></p>
SCM Branch	<p>If you have already created a branch, you can specify its location here.</p> <p>Creation of branches and merging is not supported.</p>
Version System Admin Class	<p>By default the class is provided for you.</p> <code>com.blazesoftware.repository.scm.wvcm.NdFileWVCMVersionSystemAdmin</code>
System Version Manager Class	<p>By default the class is provided for you.</p> <code>com.blazesoftware.repository.scm.wvcm.cc.NdccVersioningSystemVersionManager</code>

Repository Type Connection Parameter	Description
SCM Workspace Folder	The path to the root of your ClearCase view directory. Used for repository creation and is an alternative to creating a temporary workspace. Example: C:/ccview
Repository Home Folder	The folder inside the view where the VOB is located. This is considered to be the path for repository root. Example: /vobs/myVob
ScmRepositoryName	Name of the repository you want to create in the VOB. Eventually when the repository is created, the full path to the repository location refers to repository root and repository name. Example: repo1
SCM User	User name for the server connection.
SCM Password	Password for the server connection.

- 6 Click **OK**.

Using the External SCM Versioning System Outside the Blaze Advisor Context

The Blaze Advisor repository requires that the attribute file (.innovator_attbs) and the corresponding item are in sync with respect to their version history. If your external SCM versioning system assigns separate version numbers for the attribute file and the item, as in the case of ClearCase and CVS, then you need to make sure that you keep the two files in sync. If the content of only one of these files changes, you need to check in the other file even if it has not changed in order to make sure that the synchronization requirement does not get violated. If an item and its corresponding attribute are not in sync with respect to their version history, this may result in viewing or other problems in Blaze Advisor.

Implementing a Database Repository

You can create a Database repository in the FICO® Blaze Advisor® decision rules management system to store your projects, files, and folders. The repository is stored in a table you create on a relational database management system (RDBMS) on a server. The Database repository can be configured with or without a Blaze Versioning (BVS) service that uses a private workspace.

There are several tasks you need to complete prior to creating a Database repository:

- “[Databases Supported by Blaze Advisor](#)” on page 286
- “[Setting Your Classpath for the Database Driver Class](#)” on page 286
- “[Creating a Database Table](#)” on page 287

After you have created your table and completed the other tasks, you can create a Database repository using the Blaze Advisor IDE or the NdRomAdminUtil, see:

- “[Creating a Database Repository Using the Blaze Advisor IDE](#)” on page 291
- “[Creating a Database Repository Using the Utility](#)” on page 296

If you need to create a workspace, a connection to an existing Database repository or a connection to a workspace, see “[Creating Workspaces or Workspace Connections for your Database Repository](#)” on page 305.

After you have created a Database repository, if you want to write an external connection class to connect to it, see “[Connecting to a Database Repository by Using an External Connection](#)” on page 306.



Important Users having direct access to the database need to be authenticated and trusted users. Any custom application accessing that database should either not access the database table used to store the Blaze Advisor repository or if the custom application does access that table, the application should ensure that all users of that application are authenticated and that the data is validated before it is written to the database table.

If you want to convert an existing file repository to a BVS Database repository, see “[Converting a Repository to and from a Database Repository](#)” on page 307.

You can also add an Authorization Manager to your repository configuration and implement an authorization service. See “[Implementing an Authorization Manager](#)” on page 389.

Databases Supported by Blaze Advisor

The following databases are supported by Blaze Advisor:

- Hypersonic HSQL
- Oracle
- IBM DB2
- Microsoft

Setting Your Classpath for the Database Driver Class

You need to set your CLASSPATH to use your JDBC driver before you can create a table to store your Database repository or use the Blaze Advisor IDE to create a Database repository connection.

The tools you use to set your CLASSPATH depend on your preferences for working with your system:

- If you prefer to launch Blaze Advisor using a command prompt, see "[Setting a CLASSPATH Using a Command Prompt](#)" on page 286.
Alternatively, you can also edit the setenv.bat file located in the <ADVISOR_HOME>/bin directory to set the CLASSPATH. If you use this method, when you launch Blaze Advisor using a command prompt, the JDBC driver will be available.
- You need to set add your driver class to the global CLASSPATH in the Blaze Advisor perspective, see "[Setting a CLASSPATH Using Blaze Advisor](#)" on page 286.

You need to set the classpath before you can use a command-line utility to create the database table that stores your repository. See "[Creating a Database Table](#)" on page 287.

Setting a CLASSPATH Using a Command Prompt

If you prefer to launch Blaze Advisor using a command prompt, you can set the CLASSPATH prior to launching Blaze Advisor.

To set the CLASSPATH using a command prompt

- 1 Open a command prompt from any location on your system.
- 2 Set the classpath to include the JDBC driver.

Example:

```
$set CLASSPATH=%CLASSPATH%;c:\<JAVA_HOME>\jdbcDriver.zip
```

Setting a CLASSPATH Using Blaze Advisor

You can add the driver class to the global CLASSPATH that will be used by the Blaze Advisor plugin you installed into Eclipse. If you do not add your driver into the Blaze Advisor Eclipse workspace, you will not be able to create your Database repository using the Blaze Advisor IDE.

To set the CLASSPATH in Blaze Advisor

- 1 With the Blaze Advisor perspective open, select **Window > Preferences**.
- 2 Select **Blaze Advisor** to open the settings for the Blaze Advisor installation.
- 3 Click **Add Library** to add the driver .jar file or any other .jar files you need to use in the Blaze Advisor IDE in the Classpath Management section and browse your local machine to add one or more .jar files to the CLASSPATH.



Tip You can select more than one holding down the SHIFT key when you make your selections.

- 4 Click **Open**.
- 5 Repeat Steps 3 and 4 for each jar file or set of jar files you need to add.

Creating a Database Table

Before you can use the Blaze Advisor IDE or the `NdRomAdminUtil` utility to create a Database repository, you need to create a table in your database to store the repository contents. Generally, you can use the `NdJdbcRepositoryUtil` utility to create a table. If you use the utility, a table with the correct fields, data types, and field sizes, is automatically created for you according to the requirements of your database. If you specified the option to add indices, those they are also added to the correct table fields. You can also use this utility to create a table if you plan to Database type workspaces. For easier maintenance, we recommend that you use a separate table to house your workspaces. See "[Running a Utility to Create a Database Table](#)" on page 287.

However, under certain circumstances you may need to create a database table manually. See "[Manually Creating a Database Table](#)" on page 289.

You can enhance the performance of operations in a Database repository by adding indices to certain columns. See "[Performance and Tuning for Databases](#)" on page 290.



Note To create a table and add the required indices in the database, you need table and index creation permissions. After you have created the database table, you need to add the following permissions to select, insert, update, and delete. These permissions allow you to create the repository in this table and allow Blaze Advisor operations.

Running a Utility to Create a Database Table

You can write a batch file or enter the arguments in a command prompt to use the `NdJdbcRepositoryUtil` utility to create a table in your database. The `com.blazesoft.repository.tools.NdJdbcRepositoryUtil` utility uses the following arguments:

Argument	Value
-createTable	This is the main argument telling the utility what you want to do.
-vendor	Possible values: <ul style="list-style-type: none">■ hsql for Hypersonic HSQL■ oracle is the default.■ ibm for IBM DB2■ odbc for Sun■ microsoft for Microsoft Access or Microsoft SQL Server■ informix for Informix■ sybase for Sybase
-driver	Database driver class Important This driver class must be in your CLASSPATH. See " Setting Your Classpath for the Database Driver Class " on page 286. The driver class is associated with your database.
-url	Database URL Example: jdbc:hsqldb:hsqldb://localhost:9001
-user	Valid user name for the Database server
-password	If used, a valid password for the user name for the Database server
-table	The name you want to use for the table.
-createIndices	Indices enhance database performance during importing, exporting, or conversions. See " Performance and Tuning for Databases " on page 290. In general, if you are using an Oracle database, indices are already created for you. However, you may still be able to take advantage of the information on performance tuning.
-f	Forces table creation. <ul style="list-style-type: none">■ If the -f flag is entered, and the table already exists, the table is dropped and then recreated.■ If the -f flag is entered, and the table does not exist, it is created.

You can write a batch file using the `com.blazesoft.repository.tools.NdJdbcRepositoryUtil` utility to create a table to store your repository:

```
java com.blazesoft.repository.tools.NdJdbcRepositoryUtil -createTable -  
vendor hsql -driver org.hsqldb.jdbcDriver -url jdbc:hsqldb:hsqldb://  
localhost:9001 -user sa -table ProjectAAARespositoryTable -createIndices -f
```



Note If you plan to copy and paste any of the existing code, please be aware that it was formatted to fit the page.

Manually Creating a Database Table

Under certain circumstances you may need to create a database table manually by running an SQL CREATE statement. If this is the case, you can create a database table with the following fields, data types, and field sizes.

Field Name	Data Type (Use data type unless otherwise noted)	Field Size	Optional (Unless specified as NOT NULL)
id	VARCHAR	40	NOT NULL, PRIMARY KEY
dirId	VARCHAR	40	NOT NULL
isItem	“BIT” Depending on the database you are using, you may need to change the data type for this field. See “ Changing the Default Data Type for a Field ” on page 290.		
name	VARCHAR	255	NOT NULL
shortDesc	VARCHAR	255	
itemType	VARCHAR	255	
templateRef	VARCHAR	255	
itemDeletedFlag	VARCHAR	255	
itemCreationId	VARCHAR	255	
lastModified	“TIMESTAMP” Depending on the database you are using, you may need to change the data type for this field. See “ Changing the Default Data Type for a Field ” on page 290.		
contentLength	“INTEGER” Depending on the database you are using, you may need to change the data type for this field. See “ Changing the Default Data Type for a Field ” on page 290.		
itemContent	“LONGBINARY” Note If you are using a field with the LONGBINARY data type, you may need to specify a size limit or increase the size limit of the field if your database uses a default value that is too low. Depending on the database you are using, you may need to change the data type for this field. See “ Changing the Default Data Type for a Field ” on page 290.		

Changing the Default Data Type for a Field

You may need to change the default data type of some of the table fields when manually creating an table in your database.

Data Type	Replace with
BIT	<ul style="list-style-type: none">■ If you are using an IBM DB2 or Informix database, use "SMALLINT" instead of "BIT".■ If you are using an Oracle database, use "NUMBER(1)" instead of "BIT".
TIMESTAMP	<ul style="list-style-type: none">■ If you are using an Oracle database, use "DATE".■ If you are using an Informix database, use "DATETIME YEAR TO FRACTION".■ If you are using a Sybase database, use "DATETIME".■ If you are using a Microsoft database, use "DATETIME".
INTEGER	If you are using an Oracle database, use "NUMBER(10)".
LONGBINARY	<ul style="list-style-type: none">■ If you are using an IBM DB2 or an Oracle database, use "BLOB".■ If you are using an Informix database, use "BYTE".■ If you are using a Sybase database, use "IMAGE".■ If you are using a Microsoft database, use "IMAGE".■ If you are using a Hypersonic database, use "LONGVARBINARY". <p>Note If you are using a field with the LONGBINARY data type, you may need to specify a size limit or increase the size limit of the field if your database uses a default value that is too low.</p>

Performance and Tuning for Databases

You can enhance the performance of operations such as importing, exporting, conversion operations, and all versioning commands such as check in, check out, cancel check out, and update status, etc., by creating an unique index for the table `id` column and a non-unique index for the directory `id` (`dir id`) column for the database table used to store a Database repository.

The following performance indices are required:

- Unique index on column "`id`"

Because this field is the primary key of the table, your database may already have placed a unique index on this column and therefore you may not need to define one explicitly.

- Non-unique index on column "`dirId`"

On some databases the following index is necessary to further improve performance:

- Non-unique index on "`dirId`" and "`itemType`"

When you create the table you may want to consider specifying table space allocation parameters and/or tune the size of the "`itemContent`" `BLOB` data type.

If you need to add indices to an existing Database repository table. Use the following SQL syntax:

```
create [unique] <index name> on <table name> (column[,...n])
```

Creating a Database Repository Using the Blaze Advisor IDE

After you have the database server connection and you have created the table, you can create a Database repository with no versioning or a Database BVS repository with private workspaces using the New Repository wizard in the Blaze Advisor IDE.

- “Database Repository With no Versioning” on page 291
- “Database Repository With a BVS Private Workspace” on page 293

You can also create the same Database repositories using the NdRomAdminUtil utility, see “[Creating a Database Repository Using the Utility](#)” on page 296.

After you exit the wizard, you need to explicitly connect to a workspace using Blaze Advisor. See “[Connecting to a Blaze Advisor Workspace or Repository](#)” on page 52.

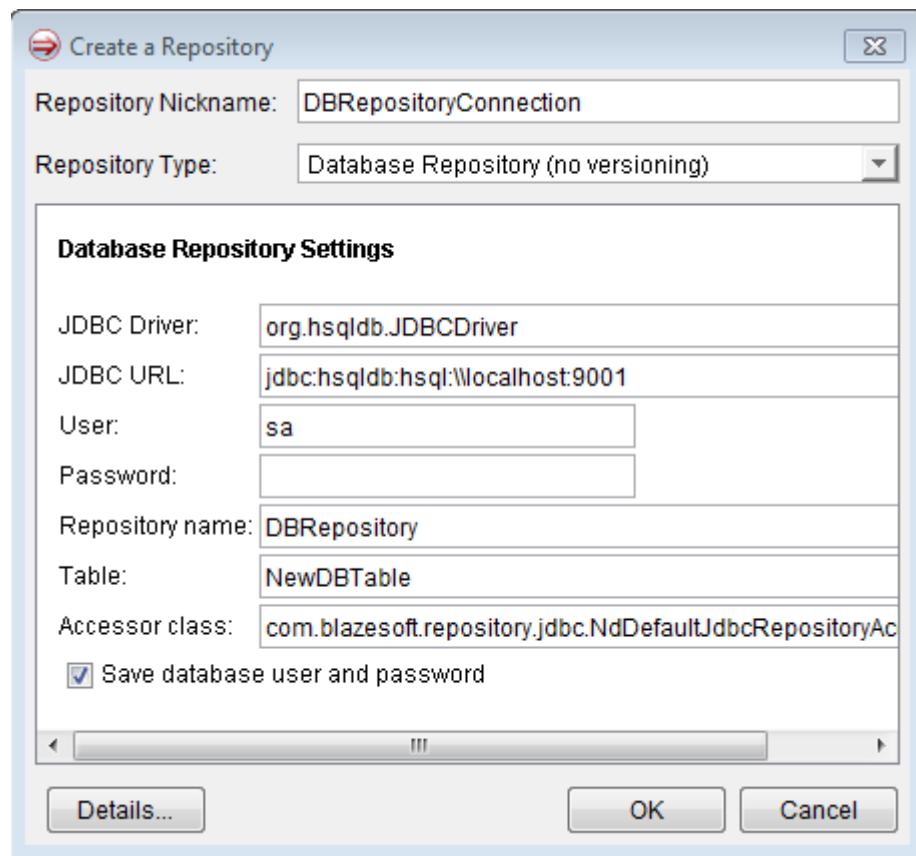
If you need to create a new workspace or a connection to an existing workspace or repository, see “[Creating Workspaces or Workspace Connections for your Database Repository](#)” on page 305.

Database Repository With no Versioning

You can create a Database repository with no versioning in Blaze Advisor by specifying the database server connection and the name of the database table you created to store your repository in the database. Below is an example of the type of information you need to enter for the connection parameters.



Important To avoid any creation errors, when entering values into the fields of the Repository Connection Parameters page, leave no trailing spaces.



To create a Database repository

- 1 Select **Repository > New Repository** command.
- 2 Select either a Database repository (no versioning) for the **Repository Type**
- 3 Enter the following information to connect to the machine that is hosting the database server:

Repository Type Parameter	Description
JDBC URL	The Database URL.
User name	Valid user name for the Database Server
Password	Valid password associated with the user name for logging onto the Database Server
Repository name	The name of the repository that will be stored in the database table on the server.

Repository Type Parameter	Description
JDBC Driver	<p>Name of the database driver This is the same driver class you used to create the database table for your repository. This driver must be included in the global CLASSPATH in the Blaze Advisor IDE. See “Setting a CLASSPATH Using Blaze Advisor” on page 286.</p>
Table	<p>Table name This is the name of the table you created using the NdJDBCRepositoryUtil utility. See “Running a Utility to Create a Database Table” on page 287.</p>
Accessor class	<p>This class is used to open a connection to the Database repository. The default class is: <code>com.blazesoft.repository.jdbc.NdDefaultJdbcRepositoryAccessor</code>.</p>

- 4 (Optional) Select the **Save database user and password** option to save the user name and password in the connection instance file and the repository configuration. See [“Saving User Name and Passwords” on page 43](#).

5 Click **OK**.

After you exit the wizard, you can connect to the repository using the Blaze Advisor IDE. See [“Connecting to a Blaze Advisor Workspace or Repository” on page 52](#).

If you want to create a new repository connection, see [“Creating Workspaces or Workspace Connections for your Database Repository” on page 305](#).

Database Repository With a BVS Private Workspace

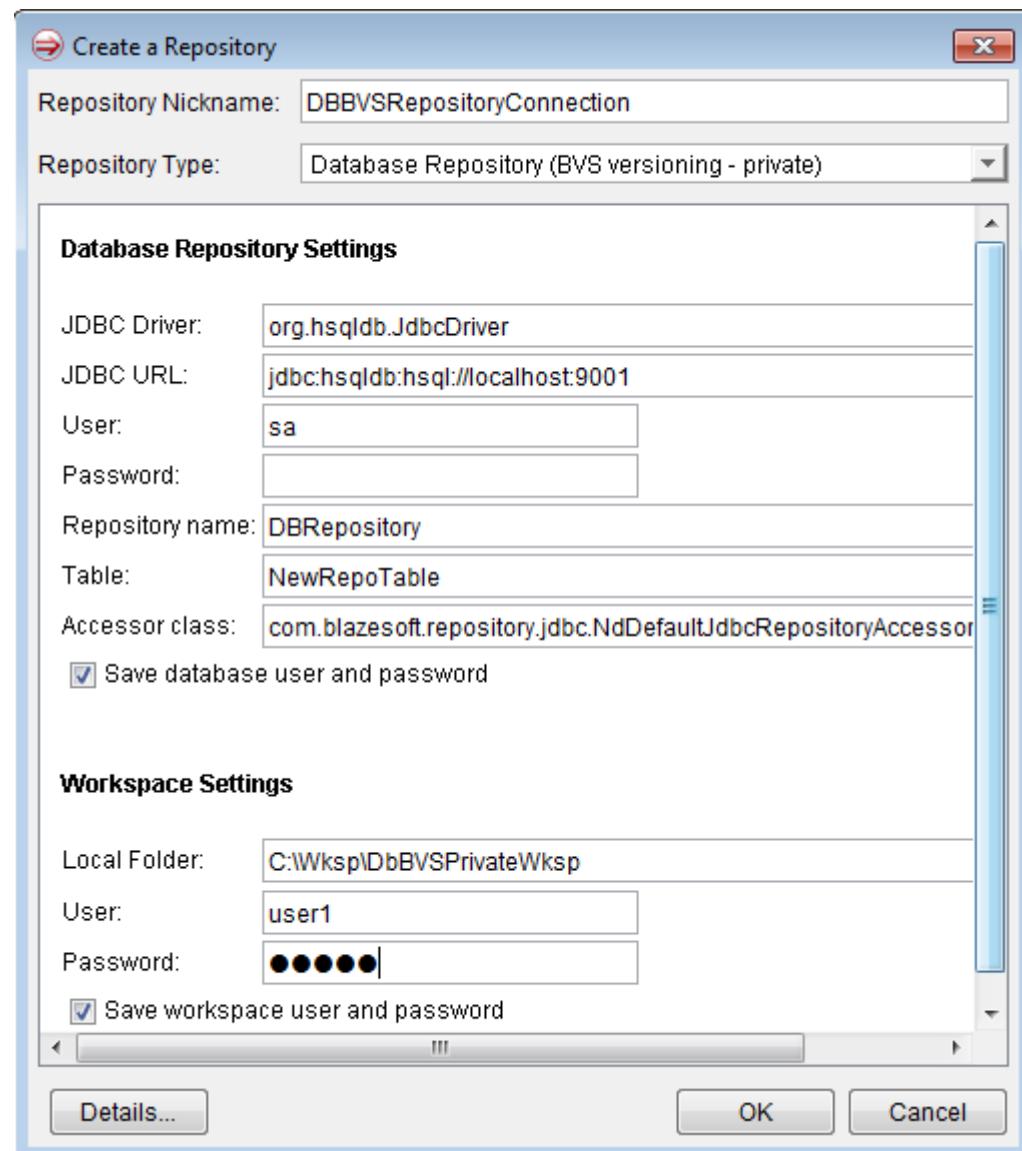
You can create a Database BVS repository with a private workspace in Blaze Advisor by specifying the database server connection and the name of the database table you created to house your repository in the database. You can also choose to save the workspace and repository connection credentials. See [“Saving User Name and Passwords” on page 43](#).

You can also set a policy to allow the workspace connection credentials to be different from the repository connection credentials. See [“Allowing Distinct Repository and Workspace Users” on page 49](#).

Below is an example of the type of information you need to enter for the connection parameters.



Important To avoid any creation errors, when entering values into the fields of the Repository Connection Parameters page, leave no trailing spaces.



To create a Database repository with a BVS private workspace

- 1 Select **Repository > New Repository** command.
- 2 Select a Database repository (BVS versioning - private) for the **Repository Type**
- 3 Enter the following information to connect to the machine that is hosting the database server:

Repository Type Parameter	Description
Local Folder	<p>The path to the private workspace.</p> <p>This is usually a folder located on your machine.</p> <p>For information about workspaces and repositories, see “The Blaze Advisor Repositories and Workspaces” on page 21.</p>
JDBC URL	The Database URL.
User name	Valid user name for the Database Server
Password	Valid password associated with the user name for logging onto the Database Server
Repository name	Name of the repository that is stored in the database table on the server.
JDBC Driver	<p>Name of the database driver.</p> <p>This is the same driver class you used to create the database table for your repository.</p> <p>This driver must be included in the global CLASSPATH in the Blaze Advisor IDE. See “Setting a CLASSPATH Using Blaze Advisor” on page 286.</p> <p>.</p>
Table	<p>Table name</p> <p>This is the name of the table you created using the NdJDBCRepositoryUtil utility. See “Running a Utility to Create a Database Table” on page 287.</p>
Accessor class	<p>This class is used to open a connection to the Database repository. The default class is:</p> <p><code>com.blazesoft.repository.jdbc.NdDefaultJdbcRepositoryAccessor</code>.</p>

- 4 (Optional) Select the **Save database user and password** option to save the user name and password in the connection instance file and the repository configuration. See [“Saving User Name and Passwords” on page 43](#).
- 5 (Optional) Select the **Save workspace user and password** option to save the user name and password in the connection instance file and the repository configuration. See [“Saving User Name and Passwords” on page 43](#).
- 6 Click **OK**.

After you exit the wizard, you can connect to the workspace using the Blaze Advisor IDE. See [“Connecting to a Blaze Advisor Workspace or Repository” on page 52](#).

After you create the repository, proceed with one or more of the following tasks:

- If you need to create a new connection to an existing workspace or a new workspace, see [“Creating Workspaces or Workspace Connections for your Database Repository” on page 305](#).

- After you have created the repository with a private workspace, you may want to verify the versioning commands, see "[Verifying Blaze Advisor Versioning Commands](#)" on page 230.

Creating a Database Repository Using the Utility

You can create a Database Repository or a Database BVS Repository with private workspaces using the `NdRomAdminUtil` utility. You need to write connection configuration files and a repository configuration file before running the utility in a command prompt and entering the command and related arguments or using a batch file.

- "[Creating a Database Repository With No Versioning Using the Utility](#)" on page 296
- "[Creating a Database BVS Repository With Private Workspaces Using the Utility](#)" on page 300

Creating a Database Repository With No Versioning Using the Utility

You can create a Database repository with no versioning services using the `NdRomAdminUtil` utility after you have established your database server connection and created the table in your database to house your repository. Before you can run the utility to create the repository, you need to write two connection configuration files and a configuration file.

- "[About Writing Files for Creating a Blaze Advisor Repository or Workspace Using the Utility](#)" on page 168
- "[Writing the Connection Configuration File for an Admin Repository](#)" on page 33
- "[Writing a Connection Configuration File for a Database Repository](#)" on page 296
- "[Writing a Repository Configuration File for a Database Repository](#)" on page 298
- "[Writing a Batch to Create a Database Repository](#)" on page 299

After you create the repository, you can use some of these files to perform other tasks using the `NdRomAdminUtil` utility, see "[The NdRomAdminUtil Utility Commands](#)" on page 162.

Writing a Connection Configuration File for a Database Repository

The repository configuration file (.cfg) you use when you run the `NdRomAdminUtil` utility determines the repository type and the available services. See "[Writing a Repository Configuration File](#)" on page 191.

The connection configuration file (.cfg) for a Database repository should contain the following structure and information:



Note For reasons of brevity, a pair of element tags will be represented by the following notation, <example/>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<RepositoryConnection/>	The RepositoryConnection tag is used when you need to create a repository connection. All other tags in this table are nested within this tag.
<Factory/>	The Factory tag is used when you want to create an instance of a class. To create an instance of a connection for a File repository, you use the fully qualified class name: com.blazesoftware.repository.jdbc.NdJdbcRepositoryConnection
<JdbcAccessor/>	The JdbcAccessor tag is used to specify the Jdbc accessor class. The default class is: com.blazesoftware.repository.jdbc.NdDefaultJdbcRepositoryAccessor.
<JdbcDriver/>	The JdbcDriver tag is used to specify the database driver class. This class must be on your classpath. See "Setting Your Classpath for the Database Driver Class" on page 286 . Example: org.hsqldb.jdbcDriver or oracle.jdbc.driver.OracleDriver
<JdbcTable/>	The JdbcTable tag is used to specify the name of the table you created in the database to house the repository. See "Creating a Database Table" on page 287 .
<JdbcUrl/>	The JdbcURL tag is used to specify the URL for the database server. Example: jdbc:hsqldb:hsq://localhost:9001 or jdbc:oracle:thin:@10.10.10.111:1521:myserver
<RepositoryName/>	The RepositoryName tag is used to specify the name of the repository that is placed in the database table on the server. This value is displayed at the root directory when you are connected to the repository in the Blaze Advisor IDE.
<User/>	The User tag is used to specify the user name for the database server connection. For example for the HSQL database server, the user name is "sa".

Tag	Description
<Password/>	The Password tag is used to specify the password for the database server connection. If no password is necessary you can omit this tag from the connection file.
<PersistCredentials/>	(Optional) Used to save the username and password to the connection instance file in the Admin Repository. Possible values: <ul style="list-style-type: none">■ false -- do not store connection credentials (default) If you choose not to store the connection information, each time you connect to the repository in the Blaze Advisor IDE, you need to manually enter your user name and password.■ true -- store the connection credentials See " Repository with BVS Versioning " on page 175.

 **Note** To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
?xml version="1.0" encoding="UTF-8" ?>
<RepositoryConnection>
    <Factory>
        com.blazesoft.repository.jdbc.NdJdbcRepositoryConnection
    </Factory>
    <PersistCredentials> true </PersistCredentials>
    <JdbcUrl>jdbc:hsql:hsqldb://localhost:9001 </JdbcUrl>
    <RepositoryName>DatabaseTestRepo</RepositoryName>
    <JdbcDriver> org.hsqldb.jdbcDriver </JdbcDriver>
    <JdbcTable> RepoTable10307 </JdbcTable>
    <JdbcAccessor>
        com.blazesoft.repository.jdbc.NdDefaultJdbcRepositoryAccessor
    <JdbcAccessor>
        <User>sa </User>
    </RepositoryConnection>
```

Writing a Repository Configuration File for a Database Repository

The following tags can be used in a configuration files for creating a file or Database repository with no versioning or authorization services:

 **Note** To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```

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

<RomConfig>
    <AllowUnusedValues> true </AllowUnusedValues>
    <RomConnectionManagerConfig>
        <ConnectionMode> 0 </ConnectionMode>
        <RomConnectionFactory>
            <JavaName>
                com.blazesoft.template.repository.impl.NdDefaultRomConnectionManager
            </JavaName>
        </RomConnectionFactory>
    </RomConnectionManagerConfig>
    <RomQueryManagerConfig>
        <RomQueryManagerFactory>
            <JavaName>
                com.blazesoft.template.repository.query.NdRomDefaultQueryManager
            </JavaName>
        </RomQueryManagerFactory>
    </RomQueryManagerConfig>
    <RomSchemaManagerConfig>
        <RomSchemaManagerFactory>
            <JavaName>
                com.blazesoft.template.repository.impl.NdDefaultRomSchemaManager
            </JavaName>
        </RomSchemaManagerFactory>
    </RomSchemaManagerConfig>
</RomConfig>

```

-  **Note** If you want to create a BVS repository with a shared workspace, you would change the value between the <AtticRepository/> tags from false to true and change the value between the <PrivateWorkspaceSupported/> tags from true to false.

Writing a Batch to Create a Database Repository

You run the `NdRomAdminUtil` utility twice to create a File BVS repository with a private workspace. First you run the utility to create the repository and then you run it a second time to create the workspace. Each time you run the utility, you must use the correct arguments and options for each task. See [“Creating a Blaze Advisor Workspace Using the Utility” on page 228](#).

To create a BVS repository with a shared workspace you run the utility once because the repository and the workspace share the same physical location.

Command, Argument, or Option	Description
createRepository	Command to create a repository
-repositoryConnection	<p>Argument takes the path to the location of the repository connection configuration file (.cfg). See “Repository Connection Configuration File Examples” on page 171.</p> <p>This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.</p>

Command, Argument, or Option	Description
-workspaceConnection Note Use this argument to create a workspace for a BVS Repository with private workspaces, File CVS Repository File Repository (Subversion versioning) or a File Repository (ClearCase versioning) using an IBM Rational ClearCase server connection.	Argument takes the path to the location of the workspace connection configuration file (.cfg) See "Writing a Connection Configuration File for a Private or a Local Workspace" on page 185 . This file contains information such as the repository name, the workspace connection name, the user name and password for the connection, and the path to the workspace.
-systemconnection	Argument takes the path to the location of the system Folder connection configuration file (.cfg). See "Repository Connection Configuration File Examples" on page 171 . This file contains information such as the name of the repository which in this case is Admin Repository, the location of the Admin Repository and the user name.
--configuration	Argument takes the path to the location of the configuration file(.cfg).See "Writing a Repository Configuration File" on page 191 .
-verbose	Option prints out all details messages into the command console.
-m	Option to add a comment when creating a repository. Note Use double quotes around your comment text.
-debug	Option prints out tracing information into the command console.

Example of a batch file to create a File BVS repository using pathnames:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil
createRepository
-repositoryConnection C:/Repo_Connections/ConnectionFiles/DefaultRepo.cfg
-systemConnection C:/Repo_Connections/ConnectionFiles/sysAdmin.cfg -
-configuration C:/Blaze/Advisor/lib/DbRepository.cfg -verbose
```

After you create the repository, you need to create a repository connection in the Blaze Advisor IDE, see ["Creating Workspaces or Workspace Connections for your Database Repository" on page 305](#).

Creating a Database BVS Repository With Private Workspaces Using the Utility

You can create a Database repository using the `NdRomAdminUtil` utility after you have established your database server connection and created the table in your database to house your repository. Before you can run the utility to create the repository, you need to write two connection configuration files and a configuration file.

- ["About Writing Files for Creating a Blaze Advisor Repository or Workspace Using the Utility" on page 168](#)
- ["Writing the Connection Configuration File for an Admin Repository" on page 33](#)

- “Writing a Connection Configuration File for a Database BVS Repository” on page 301
- “Writing a Repository Configuration File for a Database BVS Repository” on page 303
- “Writing a Batch File to Create a Database BVS Repository” on page 304

After you create the repository, you can use some of these files to perform other tasks using the `NdRomAdminUtil` utility, see “[The NdRomAdminUtil Utility Commands](#)” on page 162.

Writing a Connection Configuration File for a Database BVS Repository

You can write a connection configuration file (`.cfg`) for a file or Database repository with BVS versioning. This connection file needs to contain the structure and the values described in the table below.

You can use a shared or a private workspace with a File BVS repository and a private workspace with a Database repository. If you use a private workspace, you need to write a separate connection configuration file and run the utility a second time to create the workspace. See “[Writing a Connection Configuration File for a Private or a Local Workspace](#)” on page 185.



Note For reasons of brevity, a pair of element tags will be represented by the following notation, `<example/>`.

Tag	Description
<code><?xml version="1.0" encoding="UTF-8" ?></code>	Processing information that contains the XML version and character-encoding used.
<code><VersioningRepositoryConnection/></code>	The <code>VersioningRepositoryConnection</code> tag is used when you need to create a repository connection to a versioned repository. All other tags in this table are nested within this tag.
<code><Factory/></code>	The <code>Factory</code> tag is used when you want to create an instance of a class. To create an instance of a connection for a File repository, you use the fully qualified class name: <code>com.blazesoft.repository.jdbc.NdJdbcRepositoryConnection</code>
<code><RepositoryConnection/></code>	The <code>RepositoryConnection</code> tag is used to store the connection information to connect to the database server.
<code><JdbcAccessor/></code>	The <code>JdbcAccessor</code> tag is used to specify the Jdbc accessor class. The default class is: <code>com.blazesoft.repository.jdbc.NdDefaultJdbcRepositoryAccessor</code> .

Tag	Description
<JdbcDriver/>	<p>The JdbcDriver tag is used to specify the database driver class. This class must be on your classpath.</p> <p>See “Setting Your Classpath for the Database Driver Class” on page 286.</p> <p>Example: org.hsqldb.jdbcDriver or oracle.jdbc.driver.OracleDriver</p>
<JdbcTable/>	<p>The JdbcTable tag is used to specify the name of the table you created in the database to house the repository.</p> <p>See “Creating a Database Table” on page 287.</p>
<JdbcUrl/>	<p>The JdbcURL tag is used to specify the URL for the database server.</p> <p>Example: jdbc:hsqldb:hsq://localhost:9001 or jdbc:oracle:thin:@10.10.10.111:1521:myserver</p>
<RepositoryName/>	<p>The RepositoryName tag is used to specify the name of the repository that is placed in the database table on the server. This value is displayed at the root directory when you are connected to the repository in the Blaze Advisor IDE.</p>
<User/>	<p>The User tag is used to specify the user name for the database server connection.</p> <p>For example for the HSQL database server, the user name is “sa”.</p>
<Password/>	<p>The Password tag is used to specify the password for the database server connection. If no password is necessary you can omit this tag from the connection file.</p>
<PersistCredentials/>	<p>(Optional) Used to save the username and password to the connection instance file in the Admin Repository. If you are using BVS repository the tag is added after the closing RepositoryConnection tag.</p> <p>If you are using a BVS repository with private workspaces, the value is applied for both the repository connection and the workspace connection.</p> <p>Possible values:</p> <ul style="list-style-type: none">■ false -- do not store connection credentials (default) If you choose not to store the connection information, each time you connect to the repository in the Blaze Advisor IDE, you need to manually enter your user name and password.■ true -- store the connection credentials <p>See “Repository with BVS Versioning” on page 175.</p>

Example for a connection config for a Database repository with BVS versioning:

```

<?xml version="1.0" encoding="UTF-8" ?>
<VersioningRepositoryConnection>
    <Factory>com.blazesoft.repository.generic.version.NdNativeVersioning
        RepositoryConnection</Factory>
    <RepositoryConnection>
        <Factory>com.blazesoft.repository.jdbc.NdJdbcRepositoryConnection
        </Factory>
        <JdbcAccessor>
            com.blazesoft.repository.jdbc.NdDefaultJdbcRepositoryAccessor
        </JdbcAccessor>
        <JdbcDriver> org.hsqldb.jdbcDriver </JdbcDriver>
        <JdbcTable> BVSRepoTable10307 </JdbcTable>
        <JdbcUrl> jdbc:hsqldb:hsq://localhost:9001 </JdbcUrl>
        <RepositoryName> DatabaseBVS_Dir</RepositoryName>
        <User> user1 </User>
    </RepositoryConnection>
    <PersistCredentials>false</PersistCredentials>
</VersioningRepositoryConnection>

```

Writing a Repository Configuration File for a Database BVS Repository

You can use this repository configuration file when creating a file or Database repository with BVS versioning. The example below is for a BVS repository with a private workspace. This example is similar to the romConfig.cfg in the \lib directory of your Blaze Advisor installation. See ["Sample Connection and Configuration Files Installed With Blaze Advisor" on page 170](#).



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```

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

<RomConfig>
    <AllowUnusedValues> true </AllowUnusedValues>
    <RomConnectionManagerConfig>
        <ConnectionMode> 0 </ConnectionMode>
        <RomConnectionManagerFactory>
            <JavaName>
                com.blazesoft.template.repository.impl.NdDefaultRomConnectionManager
            </JavaName>
        </RomConnectionManagerFactory>
    </RomConnectionManagerConfig>
    <RomQueryManagerConfig>
        <RomQueryManagerFactory>
            <JavaName>

```

```
    com.blazesoftware.template.repository.query.NdRomDefaultQueryManager
  </JavaName>
</RomQueryManagerFactory>
</RomQueryManagerConfig>
<RomSchemaManagerConfig>
  <RomSchemaManagerFactory>
    <JavaName>
      com.blazesoftware.template.repository.impl.NdDefaultRomSchemaManager
    </JavaName>
  </RomSchemaManagerFactory>
</RomSchemaManagerConfig>
<RepositoryConfig>
  <AtticRepository> false </AtticRepository>
  <RepositoryVersionManagerConfig>
    <RepositoryVersionManagerFactory>
      <JavaName>
        com.blazesoftware.repository.generic.version.NdNativeRepositoryVersion
          Manager
      </JavaName>
    </RepositoryVersionManagerFactory>
    <Impersonate> true </Impersonate>
    <PrivateWorkspaceSupported> true </PrivateWorkspaceSupported>
  </RepositoryVersionManagerConfig>
</RepositoryConfig>
...
</RomConfig>
```

Writing a Batch File to Create a Database BVS Repository

You run the `NdRomAdminUtil` utility twice to create a Database BVS repository with a private workspace. First you run the utility to create the repository and then you run it a second time to create the workspace. Each time you run the utility, you must use the correct arguments and options for each task. See “[Creating a Blaze Advisor Workspace Using the Utility](#)” on page 228.

Command, Argument, or Option	Description
<code>createRepository</code>	Command to create a repository
<code>-repositoryConnection</code>	Argument takes the path to the location of the repository connection configuration file (.cfg). See “ Repository Connection Configuration File Examples ” on page 171. This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
<code>-workspaceConnection</code> Note Use this argument to create a workspace for a BVS Repository with private workspaces, File CVS Repository File Repository (Subversion versioning) or a File Repository (ClearCase versioning) using an IBM Rational ClearCase server connection.	Argument takes the path to the location of the workspace connection configuration file (.cfg) See “ Writing a Connection Configuration File for a Private or a Local Workspace ” on page 185. This file contains information such as the repository name, the workspace connection name, the user name and password for the connection, and the path to the workspace.

Command, Argument, or Option	Description
-systemconnection	Argument takes the path to the location of the system Folder connection configuration file (.cfg). See "Repository Connection Configuration File Examples" on page 171 . This file contains information such as the name of the repository which in this case is Admin Repository, the location of the Admin Repository and the user name.
--configuration	Argument takes the path to the location of the configuration file(.cfg).See "Writing a Repository Configuration File" on page 191 .
-verbose	Option prints out all details messages into the command console.
-m	Option allows you to add a comment. Note Use double quotes around your comment text.
-debug	Option prints out tracing information into the command console.

Example of a batch file to create a File BVS repository using pathnames:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil
createRepository
-repositoryConnection C:/Repo_Connections/ConnectionFiles/DefaultRepo.cfg
-systemConnection C:/Repo_Connections/ConnectionFiles/sysAdmin.cfg -
-configuration C:/Blaze/Advisor/lib/DbBVSRepository.cfg -verbose
```

After you create the repository, proceed with one or more of the following tasks:

- If you want to create a workspace using the utility, you write a connection configuration file for the workspace. See ["Creating a Blaze Advisor Workspace Using the Utility" on page 228](#).
- If you want to create a new connection to an existing workspace or a new workspace using the Blaze Advisor IDE, see ["Creating Workspaces or Workspace Connections for your Database Repository" on page 305](#).
- After you have created the repository with a private workspace, you may want to verify the versioning commands, see ["Verifying Blaze Advisor Versioning Commands" on page 230](#).

Creating Workspaces or Workspace Connections for your Database Repository

If you have created your Database repository using the Blaze Advisor IDE, you can use the connection that was created at the same time to connect to your workspace or repository in the IDE. See ["Connecting to a Blaze Advisor Repository" in *DevelopingRuleProjects.pdf*](#).

If you created your Database repository using the NdRomAdminUtil utility or if you want your users to connect to a workspace using a Database repository, you need to complete one or more of the following tasks in the Blaze Advisor IDE:

- Create a new connection to an existing workspace or repository
See “[Creating a New Blaze Advisor Workspace or Repository Connection](#)” on page 58.
- Create a new workspace
See “[Creating a Local or Private Workspace](#)” on page 226.
When you create a new workspace, you create a connection to it at the same time.

Connecting to a Database Repository by Using an External Connection

If you already have a Database repository, you can use the `com.blazesoft.repository.jdbc.NdJdbcRepositoryConnection` class to programmatically connect to it.

For more information about the `NdJdbcRepositoryConnection` class, see Help > API Reference and click on the Innovator (Frames version) or Innovator (Non-frames version) link and then click on the link for `com.blazesoft.repository.jdbc` package.

Here is an example of how to connect to a Database repository programmatically with the use of an external database connection:

```
...
//construct a JDBC repository connection
Hashtable connectionArgs = new Hashtable();
//the connection class
connectionArgs.put(NdJdbcRepositoryConnection.REPOSITORY_CONNECTION_CLASS_TAG,
    "com.blazesoft.repository.jdbc.NdJdbcRepositoryConnection")
//the repository type class
connectionArgs.put(NdRepositoryFactory.REPOSITORY_CLASS_TAG,
    "com.blazesoft.repository.jdbc.NdJdbcRepository");
//the name of the repository
//this name is stored in the database table
connectionArgs.put(NdRepository.REPOSITORY_NAME_TAG,
    "AdvisorDatabaseRepository");
//the database accessor class
//the accessor class is what the repository relies on for database interactions
//the default accessor is
//com.blazesoft.repository.jdbc.NdDefaultJdbcRepositoryAccessor
connectionArgs.put(NdJdbcRepositoryConnection.REPOSITORY_JDBC_CONNECTION_ACCESSOR
_TAG, "com.blazesoft.repository.jdbc.NdDefaultJdbcRepositoryAccessor");
//the name of the database table where the repository is contained
connectionArgs.put(NdJdbcRepositoryConnection.REPOSITORY_JDBC_CONNECTION_
TABLE_TAG, "DatabaseRepoTable");
//the external database connection where dbConnection is an instance of
//java.sql.Connection
connectionArgs.put(REPOSITORY_JDBC_NATIVE_CONNECTION_TAG, dbConnection);
//construct the JDBC repository connection
NdJdbcRepositoryConnection jdbcRepConnection=null;
try{
    jdbcRepConnection =(NdJdbcRepositoryConnection)
NdRepositoryFactory.newRepositoryConnection(connectionArgs);
    NdJdbcRepository jdbcRepository =(NdJdbcRepository)
NdRepositoryFactory.newWorkspace(jdbcRepConnection);
    jdbcRepository.openConnection(NdRepository.MODE_READ_WRITE);
```

```

    }
    catch (NdRepositoryFactoryException) e{
        e.printStackTrace();
    }
    ...

```

Converting a Repository to and from a Database Repository

There are several ways you can *convert* the contents of a repository to and from a Database repository in Blaze Advisor. When you use the commands in the Blaze Advisor IDE or the utility command to convert the source workspace contents connection type, you are essentially importing or exporting the contents of the source workspace to use the new Database repository connection type. For example, if you have a File repository and you want to use a Database repository, you can export the source workspace contents to a target workspace created for a Database repository connection. From that point, you can connect to the Database repository and continue working on the files in the new workspace. Likewise, if you have a File repository and you want to use a Database BVS repository, you can use the same commands in the Blaze Advisor IDE or the utility command to convert your workspace created in the File repository to use a Database BVS repository connection.

Both the source and target workspace must already exist for you to convert the workspace contents successfully. In the Blaze Advisor IDE, you can use either the File > Export or File > Import command depending on whether you have the source or target workspace open. If you have the source workspace open and you want to convert to another repository, you would use the File > Export > Export current workspace to another repository command. If you have the target workspace open, you would use the File > Import > Import from another repository command. If you use the utility, you use the `exportImportWorkspace` command and reference the source and target workspaces.

You can create a BVS Database repository with private workspaces. You can move the contents of the source workspace to the directory location of the private workspace.

If you have a File CVS repository and you want to use a Database BVS repository with a private workspace, assuming that both the source and target workspaces are file type, you can move the contents from your source workspace to the target workspace and check in the changes. If the workspace types are not both file, you may need to convert the workspace(s) to be a file type before proceeding with the conversion. Please note that the version history is not persisted to the target repository connection.

You can use one of these methods to move your workspace contents:

- “Importing a Blaze Advisor Contents from Another Blaze Advisor Workspace” in *DevelopingRuleProjects.pdf*
- “Exporting a Blaze Advisor Contents to Another Blaze Advisor Workspace” in *DevelopingRuleProjects.pdf*
- “[Exporting and Importing a Workspace Using the Utility](#)” on page 75

Implementing MongoDB Repositories

The Blaze Advisor MongoDB repositories available in the FICO® Blaze Advisor® decision rules management system provide scalable shared storage that allows you to deploy your RMAs or rule service deployments on the cloud.

There are several combinations of MongoDB repositories supported by Blaze Advisor to accommodate the need for cloud compatibility and to provide versioning and authorization services.

Rule Development Tasks	Administrative Tasks
"MongoDB and Blaze Advisor Repositories" on page 309	"Installation and Setup of the MongoDB Database" on page 310
"Blaze Advisor MongoDB Repository Configurations" on page 322	"MongoDB Authentication and Blaze Advisor Repositories" on page 314.
"Creating a MongoDB Repository Using the Blaze Advisor IDE" on page 327	<ul style="list-style-type: none"> ■ "Importing a Blaze Advisor Workspace or Repository Connection" on page 61 ■ "Exporting a Blaze Advisor Workspace or Repository Connection" on page 62
"Creating a MongoDB Repository Using the NdRomAdminUtil Utility" on page 338	"Converting to a Cloud Compatible MongoDB Repository" on page 381
"RMA Generation for MongoDB Repositories and Workspaces" on page 385	For more information about administrative tasks for the MongoDB, see http://docs.mongodb.org/manual/administration/



Note If you are upgrading your existing MongoDB database to a new version of MongoDB server, consult the MongoDB documentation.

For more information about the MongoDB database, see <http://www.mongodb.org/>

MongoDB and Blaze Advisor Repositories

MongoDB is a NoSQL document database that provides for high performance, accessibility and scalability. MongoDB and MongoDB Enterprise Edition are supported by FICO OpenShift and other cloud platforms. If you want to know if the your cloud platform supports MongoDB, please consult the MongoDB documentation at <http://docs.mongodb.org/ecosystem/platforms/>

Each Blaze Advisor MongoDB repository stored in a MongoDB database as a collection not a table. Collections are groupings of document-based, JSON (Javascript Object Notation)-style objects. Each entry (row) inside the same collection is a document and

can have a different number of fields, a different structure, and common fields in a collection's documents may hold different types of data. If you create a Blaze Advisor MongoDB Repository (BVS versioning - private), the workspaces are also stored as MongoDB databases. If you create a Blaze Advisor MongoDB Repository (BVS versioning - private local file workspace), the workspaces are stored as file workspaces.

Each MongoDB database contains a `system.users` collection that stores all the credentials for the database. In addition, the MongoDB database has an admin database where you can specify administrator roles and privileges for all databases or for a specific database. See "[RMA Generation for MongoDB Repositories and Workspaces](#)" on page 385.

Before you can create a Blaze Advisor repository or workspace in the MongoDB database, you need to install MongoDB either on your local or remote machine, see "[Installation and Setup of the MongoDB Database](#)" on page 310.

The MongoDB database can be started from a command line as the primary database process that runs on an individual server, see "[Running the MongoDB Database](#)" on page 313.

The MongoDB database can also be run as a Windows service. See "[Running the MongoDB Database as a Service on Windows](#)" on page 314.

You can also set up the administrative credentials using the shell, see "[Using the Mongo Shell](#)" on page 320.

Installation and Setup of the MongoDB Database

In order to use a Blaze Advisor MongoDB repository, you need to install the MongoDB database or have access to a machine with the MongoDB database installed.

- "Versions Supported" on page 310
- "Opening the MongoDB Port" on page 311
- "Closing the Default Port" on page 311
- "Installing the MongoDB Database" on page 311
- "Creating the Data Folder" on page 312
- "Creating a Log Folder" on page 312
- "Configuring the MongoDB Database" on page 313
- "Running the MongoDB Database" on page 313
- "Running the MongoDB Database as a Service on Windows" on page 314

Versions Supported

For a complete listing of the supported MongoDB clients and drivers on Windows and Linux, please see "Supported Databases" in *InstallationAndSetup.pdf*.

Opening the MongoDB Port

If you are running on Windows, you need to explicitly open the default MongoDB port on the machine where the MongoDB database will be running. On other operating systems, it may also be necessary to open ports if they have firewalls installed as Windows does. You need to follow the procedures for the firewall on that operating system. We provide a convenient batch script for the default MongoDB port on Windows firewall. Other ports may need to be opened for more complex MongoDB setups (see the MongoDB documentation for more information).

You also need to open the port on the client machines that will connect to the machine running the database. You must run this batch file in a command prompt in Run As Administrator mode.

To open the default MongoDB port

- 1 Select the **Start menu > All Programs > Accessories** and then right-click the Command Prompt to select **Run as Administrator**.
- 2 Change the directory to <ADVISOR_HOME>/bin and enter `registerMongoDBSupport.bat`.

Closing the Default Port

If you are running on Windows and you want to close the default MongoDB port on the machine where MongoDB is running. A batch file is provided for you to close the port. You must run this batch file as an administrator.

To close the default MongoDB port

- 1 Select the **Start menu > All Programs > Accessories** and then right-click the Command Prompt to select **Run as Administrator**.
- 2 Change the directory to <ADVISOR_HOME>/bin and enter `unregisterMongoDBSupport.bat`.

Installing the MongoDB Database

You can download a MongoDB installation package from <http://www.mongodb.org/downloads>.

The MongoDB installation is self-contained and does not have any other system dependencies. You can install and run the MongoDB from any folder you choose, however, the instructions that follow assume that you have installed it at the C:\ drive.

Most of the following instructions are similar to those that can be found on the MongoDB website, however, we have included some steps tailored specifically for creating and connecting the Blaze Advisor MongoDB repositories.

To install MongoDB

- 1 From the Downloads directory, extract the archive to the C:\ Drive.
- 2 Select the **Start menu > All Programs > Accessories** and then right-click the Command Prompt to select **Run as Administrator**.

- 3 Enter the following in the command prompt:

```
cd \  
move C:\mongodb-win32-* C:\mongodb
```

The location at C:\mongodb is known as <MONGODB_HOME>.

- 4 Next you create a **data** folder.

See "[Creating the Data Folder](#)" on page 312.

Creating the Data Folder

The MongoDB requires a data folder to store its files. The data folder is where the repository and workspace (if applicable) are stored. The instructions on the MongoDB website includes instructions on creating a data folder at the default location, C:\data. However, we recommend that you add it to your MongoDB installation.

To create a data folder

- 1 Select the **Start menu > All Programs > Accessories** and then right-click the Command Prompt to select **Run as Administrator**.
- 2 Change the directory to <MONGODB_HOME>.

- 3 Enter the following command:

```
md data  
md data\db
```

The result should be similar to: C:\mongodb\data\db. This location is known as the dbpath.

You can create a log folder. See "[Creating a Log Folder](#)" on page 312.

Creating a Log Folder

We recommend that you create a log folder in your <MONGODB_HOME> to contain the log of database server activity.

To create a log folder

- 1 Select the **Start menu > All Programs > Accessories** and then right-click the Command Prompt to select **Run as Administrator**.
- 2 Change the directory to <MONGODB_HOME>.

- 3 Enter the following command:

```
md log
```

The result should be similar to: C:\mongodb\log. This location is known as the logpath.

Configuring the MongoDB Database

You create a configuration (.cfg) file for the MongoDB database and store it in the <MONGODB_HOME>. The configuration file includes information such as whether or not authentication is used, if there is a log used and if the log can be appended, and the storage location for the data. The configuration file we create includes a small subset of all the configuration options available. For more information about the configuration options, see
<http://docs.mongodb.org/manual/reference/configuration-options/>

To configure the MongoDB database

- 1 Open a text editor and enter the following:

- `#auth=true`

By default, authentication is false. You add this option to the configuration file so that in future when you want to make your environment more secure you can add authentication by uncommenting this option. See “[RMA Generation for MongoDB Repositories and Workspaces](#)” on page 385.

- `logpath=<MONGODB_HOME>\log\mongo.log`

Sets the location of the log directory and create the mongo.log file.

We recommend that you create a log file to record all server activity for troubleshooting purposes.

If you do not specify a log file, the server activity will display in the command line if you manually start and stop the database server. You also do not need to specify the `logappend` option.

- `logappend=true`

Sets the way that log information is recorded to append so that each log report is appended to the end of the file.

- `dbpath=<MONGODB_HOME>\data\db`

Sets the location of where the database files are stored. If the default (C:\data\db) was used we would not have to set the dbpath.

- 2 Save the file as mongod.cfg in the <MONGODB_HOME> location.

Running the MongoDB Database

You can start a MongoDB database from a command line by using the `mongod` command and specifying options.

To run an MongoDB instance

- 1 Select the **Start menu > All Programs > Accessories** and then right-click the Command Prompt to select **Run as Administrator**.
- 2 Change your directory to your <MONGODB_HOME>\bin.
- 3 Enter `mongod --config <MONGODB_HOME>\mongod.cfg`
If you open the `mongo.log` in a text editor, you see a log of all server activity.

Running the MongoDB Database as a Service on Windows

If you are running MongoDB database on Windows, you may find it convenient to set it up as a Windows service and avoid the need to manually run MongoDB from the command line. Instead, MongoDB automatically starts in the background when you log onto your computer.

To run an MongoDB database as a Service on Windows

- 1 Select the **Start menu > All Programs > Accessories** and then right-click the Command Prompt to select **Run as Administrator**.
- 2 Change your directory to your <MONGODB_HOME>\bin.
- 3 Enter `mongod --config <MONGODB_HOME>\mongod.cfg --install`
- 4 To start the MongoDB service the first time, enter
`net start MongoDB`
- 5 To stop the MongoDB service, enter
`net stop MongoDB`

Hereafter, you do not need to start the MongoDB service manually using the net start command. It will start automatically when you start your computer. To see a log of the service activity, you can open the mongo.log file. See “[Creating a Log Folder](#)” on page 312.

You can also start and stop the Mongo DB service from Start > Control Panel > Administrative Tools > Services control panel.

MongoDB Authentication and Blaze Advisor Repositories

The MongoDB database can be run with or without authentication enabled via the auth configuration option. By default, the MongoDB database is initially configured to run without authentication.

This section contains the following topics:

- “[MongoDB URL Connection Parameter](#)” on page 315
- “[MongoDB Authentication Type Connection Parameter](#)” on page 315
- “[Connecting to MongoDB in Non-authentication Mode](#)” on page 316
- “[Connecting to MongoDB in Authenticated Mode](#)” on page 316
- “[Creating Authorized Users](#)” on page 317
- “[Using the Localhost Exception](#)” on page 317
- “[Using Administrative Credentials](#)” on page 319
- “[Using the Mongo Shell](#)” on page 320
- “[Creating an Administrative User](#)” on page 321

MongoDB URL Connection Parameter

The format for what you enter in the MongoDB URL for Blaze Advisor MongoDB repositories have the same general format as the MongoDB URI as described in the MongoDB documentation with the following exceptions:

- **mongodb:**
This prefix is optional.
- **/database**
This component should not be used.
- **username:password@**
This component should not be used.
- You can also use MongoDB connection string options to modify the behavior of your connection.

For example if you wanted to change the connection time out to 10000 microseconds and require that write operations commit the data to the on disk journal before the operation returns, you would add the following options to your MongoDB URL separated by an ampersand (i.e. &):

```
<hostName>:<portNumber>/?connectTimeoutMS=10000&journal=true
```

- You can use 127.0.0.1 or the IP address of your machine if you install the MongoDB database on your local machine. If you are connecting to the MongoDB database on a remote machine, you would use the IP address of that machine.
- The default MongoDB port number is 27017. Typically, you would use this port number if the MongoDB URL consists of only a single host name. For information on running the utility to open the port 27017, see ["Opening the MongoDB Port" on page 311](#).
- The general form of the MongoDB URL that can be entered for a Blaze Advisor MongoDB repository connection is:
`host1[:port][,host2][:port2]...[,hostN][:portN]][?options]`
For example:

```
127.0.0.1:27017/?connectTimeoutMS=10000&journal=true
```

MongoDB Authentication Type Connection Parameter

You can connect to the MongoDB database using one of the following MongoDB authentication types. The authentication type you choose depends on the authenticated mode of the MongoDB database you want to use for your connection.

- If you are creating or connecting to a MongoDB Repository (no versioning) by default the authentication type for this configuration is None.
- If you are creating or connecting to a MongoDB Repository with a versioning service or an authorization service, you need to ask your MongoDB administrator, if the MongoDB database you are using is running in authenticated mode or non-authenticated mode before selecting an authentication type.

The following authentication types are available for most MongoDB Repository configurations:

- **None**

The default setting.

See "[Connecting to MongoDB in Non-authentication Mode](#)" on page 316.

- **MongoDB**

If you are running the MongoDB database configured with authentication mode enabled, you must use this option. See "[Connecting to MongoDB in Authenticated Mode](#)" on page 316.

Connecting to MongoDB in Non-authentication Mode

If you are running the MongoDB database in non-authenticated mode, you can connect to it using the None authentication type in the Create a Repository wizard. By default, the MongoDB repository (no versioning) configuration uses the None authentication type.

Running MongoDB in non-authenticated mode and the None authentication type can be convenient when setting up new repositories or workspaces or adding new users however it can only be safely used locally on one machine or on secure networks where everyone is a trusted user. For information on setting up new users using localhost, see "[Using the Localhost Exception](#)" on page 317.

When you use the None authentication type, no MongoDB authentication is used when connecting to the MongoDB database. However, keep in mind that if you are connected to a Blaze Advisor repository with a versioning service or authorization service, Blaze Advisor authentication is still used to authorize users for access to items within the repository or workspace.

When a MongoDB database is running without authentication enabled, you are able to do the following:

- Create new repositories and workspaces in the MongoDB database remotely or locally (via localhost) and the None authentication type.
- Connect to Blaze Advisor repositories in the MongoDB database either remotely or locally (via localhost) and the None authentication type.
- Create authorized users for later use when running MongoDB with authentication enabled by connecting locally (via localhost) with the authentication type set to MongoDB. See "[Creating Authorized Users](#)" on page 317.

Connecting to MongoDB in Authenticated Mode

If you are running the MongoDB database in authenticated mode, you can connect to it using the MongoDB authentication type and the MongoUrl using the IP address of the machine with the default port in the Create a Repository wizard.



Note MongoDB 2.4 added support for using Kerberos authentication in addition to MongoDB authentication. Blaze Advisor does not currently support Kerberos authentication.

When MongoDB is running in authenticated mode, the MongoDB authentication type must be used when connecting to the MongoDB database from Blaze Advisor.

If you create a MongoDB repository with a set of registered users and configured with an authorization manager or a versioning service in a MongoDB database running in authenticated mode, when users attempt to connect to the repository they will be authenticated by the MongoDB database. When they attempt to access items within the repository or workspace the authorization manager or versioning service configured with the repository controls their access. Therefore, the user names and passwords for registered users of the database need to be the same as those used by the authorization manager class otherwise the authentication will fail.

If you create a MongoDB repository configured with an authorization manager or a versioning service in a MongoDB database running in non-authenticated mode when users attempt to connect to the repository the only authentication available is through the authorization manager or versioning service configured with the repository.

Prior to running MongoDB with authentication enabled, you can create a list of authorized users for your Blaze Advisor MongoDB repository and workspaces using one of the methods. ["Creating Authorized Users" on page 317](#).

Creating Authorized Users

For your users to have access to the Blaze Advisor MongoDB repository, they need to be added to the system.user collection. The `system.users` collection in each repository hosted in MongoDB contains the credentials for all users. You can add to the `system.users` collection for your repository or workspace in the Blaze Advisor IDE using the following methods.

- ["Using the Localhost Exception" on page 317](#)
- ["Using Administrative Credentials" on page 319](#)
- ["Using the Mongo Shell" on page 320](#)
- ["Creating an Administrative User" on page 321](#)

Using the Localhost Exception

When you are initially setting up your repository and your users, you can use what is known as the localhost exception to create a new Blaze Advisor repository or workspace, to connect to a Blaze Advisor repository or to add a user to an existing Blaze Advisor repository or workspace, previously created with authentication disabled. You can use the localhost exception if you connect via localhost when the MongoDB database is running in authenticated mode and you do not have any administrative users defined in the MongoDB admin database, and localhost is the only host in the URL. The Repository Settings fields described in the steps below are available in the Create a Repository wizard, Connect to Repository wizard, or New Workspace wizard in the Blaze Advisor IDE.



Note The localhost exception is not needed with the MongoDB repository (no versioning) as this configuration by default uses the None authentication type.

To use the localhost exception

- 1 In the **MongoDB URL** field, enter **localhost**.
- 2 (Optional) Enter the default port number :**27017** next to **localhost** in the MongoDB URL field.

- 3 Select **MongoDB** radio button for the MongoDB authentication type.

- 4 In the **User** and **Password** fields, enter the user credentials for the repository.

If you are creating users and you are using a repository with a versioning or authorization service, you need to create user credentials that correspond to the credentials used for those services.

- 5 In the **Repository name** field, enter the name of the new or existing repository.

The name you specify here corresponds to the name used for the repository database and therefore cannot contain the following characters "/" , "\", "", "*", "<", ">" , ":" , "|" , "?" , or double quotes.

- 6 If you are adding users to a MongoDB Repository (BVS versioning - private):

- a In the **Workspace Name** field, enter the name of the workspace you want this user to access.

The name you specify here corresponds to the name used for the workspace database and therefore cannot contain the following characters "/" , "\", "", "*", "<" , ">" , ":" , "|" , "?" , or double quotes.

- b In the **User** and **Password** fields, enter the user credentials for the workspace.

If you are creating users and you are using a repository with a versioning or authorization service, you need to create user credentials that correspond to the credentials used for those services.

- 7 Click **OK**.

The user credentials you specified for the repository or workspace are added to the list of authorized users for the Blaze Advisor MongoDB repository and/or workspace hosted in MongoDB.



Note Another quick way to add users is to use an existing Blaze Advisor repository connection and modify the connection to use localhost and enable MongoDB authentication. Then you can just enter the user and password (you do not even need to do that if you previously enabled persisted credentials). When you connect with this modified connection the user credentials will be added to the MongoDB database for the repository and/or workspaces specified in the connection.

Once you have completed this your users can connect to the repository locally or remotely by entering the host name or IP address and port number in the MongoDB URL field for the repository connection and by selecting the MongoDB authentication type.

Using Administrative Credentials

If you cannot connect to the MongoDB database via localhost and/or administrative users have been defined in your MongoDB admin database, you can run the MongoDB database with authentication enabled and use the administrative user credentials to create a new repository, workspace, connect to a Blaze Advisor repository or to add a user to an existing repository or workspace, previously created with authentication disabled in the Blaze Advisor IDE. The Repository Settings fields described in the steps below are available in the Create a Repository wizard, Connect to Repository wizard, or New Workspace wizard.

For information about creating an administrative user, see ["Creating an Administrative User" on page 321](#).

To use administrative credentials

- 1 In the MongoDB URL field, enter the host name or IP address and port number.
Example: myHostMachine:27017 or 127.0.0.1:27017.
- 2 Select **MongoDB** radio button for the MongoDB authentication type.
- 3 In the **User** and **Password** fields, enter the user credentials for the repository.
If you are creating users and you are using a repository with a versioning or authorization service, you need to create user credentials that correspond to the credentials used for those services.
- 4 In the **Repository name** field, enter the name of the new or existing repository.
The name you specify here corresponds to the name used for the repository database and therefore cannot contain the following characters "/", "\", "", "*", "<", ">", ".", "|", "?", or double quotes.
- 5 If you are adding users to a **MongoDB Repository (BVS versioning - private)**:
 - a In the Workspace Name field, enter the name of the workspace you want this user to access.
 - b In the **User** and **Password** fields, enter the user credentials for the workspace.
If you are creating users and you are using a repository with a versioning or authorization service, you need to create user credentials that correspond to the credentials used for those services.
- 6 Click **OK**.
- 7 If you are prompted for administrative credentials, enter the appropriate MongoDB administrative credentials.
 - You will not be prompted for administrative credentials if the user specified in the connection already exist as an authorized user of the repository or workspace.
 - However, if the user specified in the connection already exist as authorized users but you accidentally enter an incorrect password, you see the prompt and the

connection ultimately fails with an unauthorized user message because the password was incorrect. The prompt displays in this case because without the administrative credentials it is not possible to determine the reason for the failure.

- If you cancel when prompted for administrative credentials, the connection will fail with an unauthorized user message.
- If your administrative credentials are accepted and the users do not already exist as authorized users, the user credentials you specified for the repository or workspace are added to the list of authorized users for the Blaze Advisor MongoDB repository and/or workspace hosted in MongoDB.

Your users can now connect to the repository locally or remotely by entering the host name or IP address and port number in the MongoDB URL field for the repository connection and by selecting the MongoDB authentication type.

Using the Mongo Shell

If you cannot connect to the MongoDB database via localhost and/or administrative users have been defined in your MongoDB admin database, you can run the MongoDB database with authentication enabled and use the administrative user credentials to create a new repository, workspace, connect to a repository to add a user to an existing Blaze Advisor repository or workspace, previously created with authentication disabled using the Mongo shell.

For information about creating an administrative user, see ["Creating an Administrative User" on page 321](#).

If you use the following shell commands and the repository or workspace you specify does not exist, it will be created and you can use the Blaze Advisor IDE to complete the creation of the repository or workspace.

To log onto the Mongo shell with Administrative Credentials

- 1 With the MongoDB running, start the Mongo shell in the admin database with the administrative credentials

```
mongo -u <adminUserName> -p <adminPassword> <hostName>/admin  
where hostName is the host machine where the MongoDB is running.
```

Example

```
mongo -u admin -p admin localhost/admin
```

- 2 Switch to the MongoDB database for the Blaze Advisor repository or workspace
> use <BlazeAdvisorRepositoryOrWorkspaceName>
where BlazeAdvisorRepositoryOrWorkspaceName is the Blaze Advisor repository or workspace. If the Blaze Advisor repository or workspace does not exist, it will be created in the MongoDB.

Example

```
> use DevelopmentRepo
```

- 3 You can use the following functions to manage your repositories, create users, and obtain information:
 - Show existing database names
>show dbs
 - Delete the MongoDB database for a Blaze Advisor repository or workspace
> use <BlazeAdvisorRepositoryOrWorkspaceName>
> db.dropDatabase()
 - Show the authorized users for a given Blaze Advisor repository or workspace
> use <BlazeAdvisorRepositoryOrWorkspaceName>
> db.system.users.find()
 - Add an authorized user for a given Blaze Advisor repository or workspace
> use <BlazeAdvisorRepositoryOrWorkspaceName>
> db.createUser("username" "password")
 - Remove an authorized user for a given Blaze Advisor repository or workspace
> use <BlazeAdvisorRepositoryOrWorkspaceName>
> db.removeUser("username")

Creating an Administrative User

If no users have been created in the MongoDB admin database, you can connect to the MongoDB with authentication enabled and have administrative privileges via localhost or you can connect while MongoDB is configured with authentication disabled. You can use the Mongo shell to create an administrative user with the `userAdmin` role. After you have created an admin user, you need to restart MongoDB with authentication enabled (if it is not already enabled).



Note The `db.AddUser` command that has been used in previous releases of MongoDB has been deprecated in MongoDB and can no longer be used. Instead, you need to use `db.createUser`.

To create a user with administrative credentials

- 1 Run the MongoDB database *without* authentication enabled.
- 2 Select the **Start menu > All Programs > Accessories** and then right-click the Command Prompt to select **Run as Administrator**.
- 3 Change your directory to your <MONGODB_HOME>\bin.
- 4 Enter `mongo` to open the Mongo shell.
`mongo localhost`
- 5 Select the admin database with the following command:
`use admin`
- 6 If you want to define a user with sufficient credentials to create users for databases in the MongoDB instance, for example, you may want administrative credentials to be entered when a user creates a new workspace in the RMA or IDE, enter the following to add a user with administrative credentials.

```
db.createUser({user: "adminUserName",_ pwd: "adminPassword,  
roles:[“useAdminAnyDatabase”]})
```

where the `adminUserName` and `adminPassword` can be any name and password that you wish to use in the RMA/IDE when prompted for administrative credentials.

- 7 After you have created a user with administrative credentials you can restart the MongoDB database with authentication enable.



Note The role is optional unless you want to restrict the administrative user to specific more roles rather than a full administrator role with all permissions.

- 8 After you have the `userAdmin` role in the MongoDB admin database, you can query authenticated users in that database with the following operation:
`db.system.users.find()`

See “[Using the Mongo Shell](#)” on page 320.

Blaze Advisor MongoDB Repository Configurations

There are several Blaze Advisor MongoDB repository configurations that you can choose from based on the needs of your projects, the number of users, and whether or not cloud compatibility is a requirement.

When you create a MongoDB type repository, you can set the item content compression threshold, see “[Setting the Item Content Threshold](#)” on page 326.

Repository Configuration	Description	Cloud Compatible	Authentication Type	To create
MongoDB Repository (no versioning)	<ul style="list-style-type: none"> ■ Non-versioned. ■ Can be created on MongoDB running in non-authenticated mode. ■ Recommended for rapid development and prototyping purposes or in situations where every one is a trusted user. 	Yes	<p>None (default). Works only with MongoDB running in non-authenticated mode. See “MongoDB Authentication Type Connection Parameter” on page 315.</p>	<ul style="list-style-type: none"> ■ “Creating a MongoDB Repository (no versioning) using the Blaze Advisor IDE” on page 328. ■ “Creating a MongoDB Repository (no versioning) Using the NdRomAdminUtil Utility” on page 339.
MongoDB Repository (with Authorization manager)	<ul style="list-style-type: none"> ■ Non-versioned. ■ Can be created on MongoDB running in non-authenticated mode or authenticated mode. ■ Configured with an authorization manager class that determines access to items in the repository or workspace. 	Yes	<p>Available values:</p> <ul style="list-style-type: none"> ■ None ■ MongoDB <p>See “MongoDB Authentication Type Connection Parameter” on page 315.</p>	<ul style="list-style-type: none"> ■ “Creating a MongoDB Repository (with Authorization Manager) using the Blaze Advisor IDE” on page 329. ■ “Creating a MongoDB Repository (with Authorization manager) Using the NdRomAdminUtil utility” on page 346.

Repository Configuration	Description	Cloud Compatible	Authentication Type	To create
MongoDB Repository (versioning - private)	<ul style="list-style-type: none"> ■ Blaze Versioning Service used. ■ Can be created on MongoDB running in non-authenticated mode or authenticated mode. ■ The repository and the private workspaces for each user are stored in the MongoDB system. 	Yes	Available values: <ul style="list-style-type: none"> ■ None ■ MongoDB See “MongoDB Authentication Type Connection Parameter” on page 315.	<ul style="list-style-type: none"> ■ “Creating a MongoDB Repository (BVS versioning - private) using the Blaze Advisor IDE” on page 331. ■ “Creating a MongoDB Repository (BVS versioning - private) Using the NdRomAdminUtil utility” on page 350. ■ “Creating a Workspace for a MongoDB Repository (versioning - private) Using the Blaze Advisor IDE” on page 365. ■ “Creating a Workspace for a MongoDB Repository (versioning - private) Using the NdRomAdminUtil Utility” on page 370.

Repository Configuration	Description	Cloud Compatible	Authentication Type	To create
MongoDB Repository (versioning - shared)	<ul style="list-style-type: none"> ■ Blaze Versioning Service used. ■ Can be created on MongoDB running in non-authenticated mode or authenticated mode. ■ The repository and the shared workspace are stored in the MongoDB system. 	Yes	<p>Available values:</p> <ul style="list-style-type: none"> ■ None ■ MongoDB <p>See “MongoDB Authentication Type Connection Parameter” on page 315.</p>	<ul style="list-style-type: none"> ■ “Creating a MongoDB Repository (BVS versioning - shared) using the Blaze Advisor IDE” on page 334. ■ “Creating a MongoDB Repository (BVS versioning - shared) Using the NdRomAdminUtil utility” on page 355.

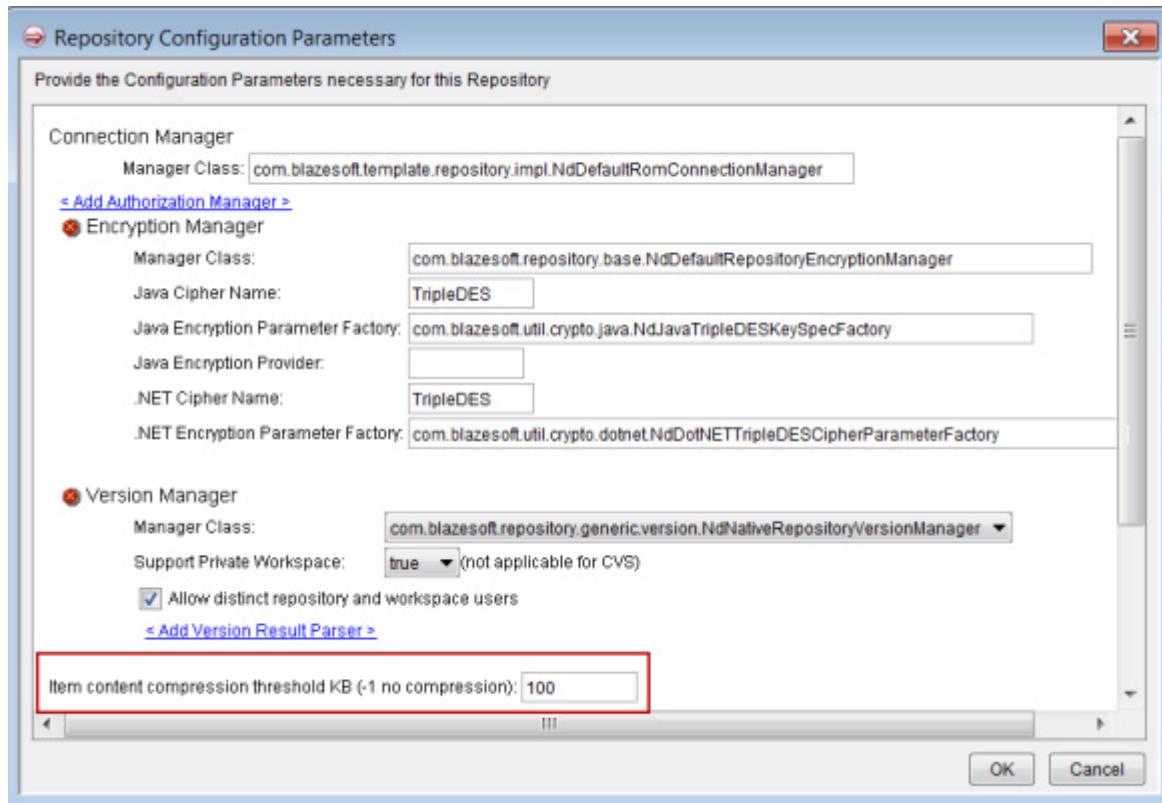
Repository Configuration	Description	Cloud Compatible	Authentication Type	To create
MongoDB Repository (versioning - private local file)	<ul style="list-style-type: none"> ■ Blaze Versioning Service used. ■ Can be created on MongoDB running in non-authenticated mode or authenticated mode. ■ The repository is stored in the MongoDB system but the file workspaces for each user are stored on a file system. 	No	<p>Available values:</p> <ul style="list-style-type: none"> ■ None ■ MongoDB <p>See “MongoDB Authentication Type Connection Parameter” on page 315.</p>	<ul style="list-style-type: none"> ■ “Creating a MongoDB Repository (BVS versioning - private local file workspace) using the Blaze Advisor IDE” on page 335. ■ “Creating a MongoDB Repository (BVS versioning - private local file workspace) Using the NdRomAdminUtil utility” on page 360. ■ “Creating a Workspace for a MongoDB Repository (BVS versioning - private local file) Using the Blaze Advisor IDE” on page 368. ■ “Creating a Workspace for a MongoDB Repository (BVS versioning - private local file) Using the NdRomAdminUtil Utility” on page 373. ■ “Converting a MongoDB Repository (BVS versioning - private local file) to a MongoDB Repository (BVS versioning - private)” on page 382.

Setting the Item Content Threshold

The <ItemContentCompressionThresholdKB/> is added to the repository configuration when you create a new MongoDB repository. This tag is used to set the threshold at which the item’s content will be stored in the repository as compressed bytes instead of as UTF-8 strings. If you have created another repository type, this tag is ignored.

If you are creating a MongoDB repository using the Blaze Advisor IDE, you see the **Item content compression threshold KB (-1 no compression)** in the Repository Configuration Parameter page that is accessed through the Details button on the Create

a Repository wizard. A compression threshold value between 100 KB and 15.9 MB can be set using this tag.



If you look in a repository configuration file, `com.blazesoftware.repository_config.cfg`, tag, the default value of the `<ItemContentCompressionThresholdKB/>` tag is -1. If you are using a MongoDB type repository and when the compression threshold is set to the default, a pre-defined internal default threshold of 100 KB is used. If the compression threshold value is set to > 16 MB, a pre-defined internal threshold value of 15.9 MB is used.

Creating a MongoDB Repository Using the Blaze Advisor IDE

A repository is a storage mechanism for projects, folders, and entities. Repositories can have one or more services that provide additional functional features such as password encryption, versioning, or authorization. There are several default Blaze Advisor repository types that come pre-configured with service that are defined in the repository configuration. There are several different MongoDB repository configurations each with a set of default and specific services.

You can create a Blaze Advisor MongoDB repository that best suits your needs by connecting to a MongoDB instance. In order to connect to the MongoDB, you need to know its URL and whether or not it is running in authenticated or non-authenticated mode.



Important When you create a Blaze Advisor MongoDB repository or workspace, the connection information is cached by the MongoDB driver in the Blaze Advisor IDE or the RMA. To avoid an error, if for any reason, you need to stop and restart the MongoDB database, exit the Blaze Advisor IDE or the RMA prior to restarting MongoDB.

You can create the following repository configurations using the Blaze Advisor IDE:

- “[Creating a MongoDB Repository \(no versioning\) using the Blaze Advisor IDE](#)” on page 328
- “[Creating a MongoDB Repository \(with Authorization Manager\) using the Blaze Advisor IDE](#)” on page 329
- “[Creating a MongoDB Repository \(BVS versioning - private\) using the Blaze Advisor IDE](#)” on page 331
- “[Creating a MongoDB Repository \(BVS versioning - shared\) using the Blaze Advisor IDE](#)” on page 334
- “[Creating a MongoDB Repository \(BVS versioning - private local file workspace\) using the Blaze Advisor IDE](#)” on page 335

After you create a MongoDB repository, if you need provide the connection for another user to connect to the repository, see “[Connecting to a MongoDB Repository Using the Blaze Advisor IDE](#)” on page 378.

Creating a MongoDB Repository (no versioning) using the Blaze Advisor IDE

You can create a MongoDB repository (no versioning) for projects where security is not the foremost concern. This repository configuration is used with MongoDB running in non-authenticated mode (the default configuration of the MongoDB).

After you create this repository, in the MongoDB database, any user who knows the URL and repository name can connect to the repository and create, update, and delete projects and project items.

After you create your repository, if you want to provide the connection information to other users, see “[Exporting a Blaze Advisor Workspace or Repository Connection](#)” on page 62.

To create a MongoDB repository

- 1 Select **Repository > New Repository**.
- 2 In the Create a Repository wizard, enter a name in the **Repository Nickname** field. This name is used to identify the repository connection in the IDE.
- 3 Select the **MongoDB Repository (no versioning)** from the Repository Type dropdown.
- 4 Enter the **MongoDB URL** for the machine where the MongoDB database is running and enter a port number using the format <machineName or IP

`address>:<portNumber>`. If you use the default MongoDB port number, entering it in the URL field is optional.

For example, `127.0.0.1:27017`.

See “[MongoDB URL Connection Parameter](#)” on page 315.

- 5 (Optional) Select **Use SSL (Requires MongoDB Enterprise Edition)**.

See “[SSL Connections and MongoDB Repositories](#)” on page 383.

- 6 Enter a name in the **Repository name** field.

The name you specify here corresponds to the name used for the repository database and therefore cannot contain the following characters “/”, “\”, “.”, “*”, “<”, “>”, “:”, “|”, “?”, or double quotes.

- 7 (Optional) Click **Details** to see the configuration parameters of the repository type you selected.

If you view the Details page and want to edit the configuration to add or delete a service, see “[Editing a Repository Configuration in the Blaze Advisor IDE](#)” on page 39.

- 8 Click **OK**.

Creating a MongoDB Repository (with Authorization Manager) using the Blaze Advisor IDE

You can create a MongoDB repository with an authorization service and no versioning service if you want to control access to items in your projects using a custom authorization manager class. If you have a requirement for your repository to be cloud compatible, you need to also ensure that your custom authorization manager implementation is also cloud compatible.

This repository configuration can be used with a MongoDB database instance that is running in authenticated mode with a set of registered users. It can also be used with a MongoDB database instance that is running in non-authenticated mode in which case the only authentication is the one provided by the Blaze Advisor repository configured with an authorization manager. See “[MongoDB Authentication and Blaze Advisor Repositories](#)” on page 314.

After you create your repository, if you want to provide the connection information to other users, see “[Exporting a Blaze Advisor Workspace or Repository Connection](#)” on page 62.

To create a MongoDB repository with Authorization manager

- 1 Select **Repository > New Repository**.
- 2 In the Create a Repository wizard, enter a name in the **Repository Nickname** field. This name is used to identify the repository connection in the IDE.
- 3 Select the **MongoDB Repository (with Authorization manager)** from the Repository Type dropdown.

- 4 Enter the **MongoDB URL** for the machine where the MongoDB database is running and enter a port number using the format <machineName or IP address>:<portNumber>. If you use the default MongoDB port number, entering it in the URL field is optional.

For example, 127.0.0.1:27017.

See “[MongoDB URL Connection Parameter](#)” on page 315.

- 5 Select a **MongoDB authentication type** based on the authentication mode of the MongoDB database you want to use.

■ **None**

■ **MongoDB**

See “[MongoDB Authentication Type Connection Parameter](#)” on page 315.

- 6 (Optional) Select **Use SSL (Requires MongoDB Enterprise Edition)**.

See “[SSL Connections and MongoDB Repositories](#)” on page 383.

- 7 Enter a user name in the **User** field.

You need to ensure that the password you enter here is the same one used with your authorization service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the username you enter here is added to the system.user collection of your repository database.

- 8 Enter a password name in the **Password** field.

You need to ensure that the password you enter here is the same one used with your authorization service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the password you enter here is added to the system.user collection of your repository database.

- 9 (Optional) You can unselect the **Save user and password** check box.

■ This option is selected by default so that users can quickly and easily connect to the workspace, repository, or both by selecting a connection in the Manage Connections, Import, or Export wizards. The password is encrypted by default. See “[Saving User Name and Passwords](#)” on page 43.

■ If this option is unselected, user names and passwords are not persisted in the system, however it also means that users will be prompted whenever a connection to the workspace, repository, or both needs to be made.

- 10 Enter a name in the **Repository name** field.

The name you specify here corresponds to the name used for the repository database and therefore cannot contain the following characters “/”, “\”, “”, “*”, “<”, “>”, “.”, “!”, “?”, or double quotes.

- 11 Click **Details** to see the configuration parameters of the repository type you selected.
 - a You need to enter the fully qualified name of the Authorization manager class you want to use in the **Authorization Manager Manager Class** field.
 - b Click **OK**.

If you view the Details page and want to edit the configuration to add or delete a service, see "[Editing a Repository Configuration in the Blaze Advisor IDE](#)" on page 39.
- 12 Click **OK**.

Creating a MongoDB Repository (BVS versioning - private) using the Blaze Advisor IDE

You can create a MongoDB Repository (BVS versioning - private) if you want to use a versioning service with your cloud compatible repository with no local file IO. In this case, the repository and the private workspaces for each user are hosted by the MongoDB database.

This repository configuration can be used with a MongoDB database instance that is configured in authenticated mode with a set of registered users. It can also be used with a MongoDB database instance that is configured in non-authenticated mode in which case the only authentication is the one provided by the Blaze Advisor repository configured with the versioning service. See "[MongoDB Authentication and Blaze Advisor Repositories](#)" on page 314.

After you create your repository, if you want to provide the connection information to other users, see "[Exporting a Blaze Advisor Workspace or Repository Connection](#)" on page 62.

To create a MongoDB repository (versioning - private)

- 1 Select **Repository > New Repository**.
- 2 In the Create a Repository wizard, enter a name in the **Repository Nickname** field. This name is used to identify the repository connection in the IDE.
- 3 Select the **MongoDB Repository (versioning - private)** from the Repository Type dropdown.
- 4 Enter a connection name for your workspace. This name is used to identify the workspace connection in the IDE.
- 5 Click **Next**.
- 6 On the MongoDB Repository Settings section, enter the connection settings for the repository:
 - a Enter the **MongoDB URL** for the machine where the MongoDB database is running and enter a port number using the format <machineName or IP

`address>:<portNumber>`. If you use the default MongoDB port number, entering it in the URL field is optional.

For example, `127.0.0.1:27017`.

See “[MongoDB URL Connection Parameter](#)” on page 315.

- b** Select a **MongoDB authentication type** based on the authentication mode of the MongoDB database you want to use.

- **None**
- **MongoDB**

See “[MongoDB Authentication Type Connection Parameter](#)” on page 315.

- 7** (Optional) Select **Use SSL (Requires MongoDB Enterprise Edition)**.

See “[SSL Connections and MongoDB Repositories](#)” on page 383.

- c** Enter a user name in the **User** field.

You need to ensure that the user name you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the user name you enter here is added to the system.user collection of your repository database.

- d** Enter a password name in the **Password** field.

You need to ensure that the password you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the password you enter here is added to the system.user collection of your repository database.

- e** Enter a name in the **Repository name** field.

The name you specify here corresponds to the name used for the repository database and therefore cannot contain the following characters “/”, “\”, “.”, “*”, “<”, “>”, “:”, “|”, “?”, or double quotes.

- f** (Optional) You can unselect the **Save database user and password** check box.

- This option is selected by default so that users can quickly and easily connect to the workspace, repository, or both by selecting a connection in the Manage Connections, Import, or Export wizards. The password is encrypted by default. See “[Saving User Name and Passwords](#)” on page 43.
- If this option is unselected, user names and passwords are not persisted in the system, however it also means that users will be prompted whenever a connection to the workspace, repository, or both needs to be made.

- 8 In the Workspace Settings section enter the workspace connection information:
 - a Enter a name in the **Workspace Name** field.

The name you specify here corresponds to the name used for the workspace database and therefore cannot contain the following characters "/", "\", ".", "*", "<", ">", ":" , "|", "?", or double quotes.
 - b Enter a user name in the **User** field.

You need to ensure that the user name you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the user name you enter here is added to the system.user collection of your repository database.
 - c Enter a password name in the **Password** field.

You need to ensure that the user name you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the user name you enter here is added to the system.user collection of your repository database.
 - d (Optional) You can select the **Save workspace user and password** check box to use a link to quickly and easily connect to the workspace or repository in Blaze Advisor. See "[Saving User Name and Passwords](#)" on page 43.
- 9 Click **Details** to see the configuration parameters of the repository type you selected.
 - a (Optional) You can allow distinct repository and workspace users, see "[Allowing Distinct Repository and Workspace Users](#)" on page 49.
 - b Click **OK**.

If you view the Details page and want to edit the configuration to add or delete a service, see "[Editing a Repository Configuration in the Blaze Advisor IDE](#)" on page 39.
- 10 Click **OK**.

If you need to create a workspace for the repository you just created, see "[Creating or Removing a MongoDB or File Workspace](#)" on page 365.

If you need to create a connection to this repository, see "[Connecting to a MongoDB Repository Using the Blaze Advisor IDE](#)" on page 378.

After you have created the repository with a private MongoDB workspace, you may want to verify the versioning commands, see "[Verifying Blaze Advisor Versioning Commands](#)" on page 230.

Creating a MongoDB Repository (BVS versioning - shared) using the Blaze Advisor IDE

You can create a MongoDB Repository (BVS versioning - shared) if you want to use a versioning service with your cloud compatible repository. In this case, the repository and the shared workspace used by all users are hosted by the MongoDB database.

This repository configuration can be used with a MongoDB database instance that is configured in authenticated mode with a set of registered users. It can also be used with a MongoDB database instance that is configured in non-authenticated mode in which case the only authentication is the one provided by the Blaze Advisor repository configured with the versioning service. See "[MongoDB Authentication and Blaze Advisor Repositories](#)" on page 314.

After you create your repository, if you want to provide the connection information to other users, see "[Exporting a Blaze Advisor Workspace or Repository Connection](#)" on page 62.

To create a MongoDB repository (versioning - shared)

- 1 Select **Repository > New Repository**.
- 2 In the Create a Repository wizard, enter a name in the **Repository Nickname** field. This name is used to identify the repository connection in the IDE.
- 3 Select the **MongoDB Repository (versioning - shared)** from the Repository Type dropdown.
- 4 Enter the **MongoDB URL** for the machine where the MongoDB database is running and enter a port number using the format `<machineName or IP address>:<portNumber>`. If you use the default MongoDB port number, entering it in the URL field is optional.
For example, `127.0.0.1:27017`.
See "[MongoDB URL Connection Parameter](#)" on page 315.
- 5 Select a **MongoDB authentication type** based on the authentication mode of the MongoDB database you want to use.
 - **None**
 - **MongoDB**See "[MongoDB Authentication Type Connection Parameter](#)" on page 315.
- 6 (Optional) Select **Use SSL (Requires MongoDB Enterprise Edition)**.
See "[SSL Connections and MongoDB Repositories](#)" on page 383.
- 7 Enter a user name in the **User** field.
You need to ensure that the user name you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the user name you enter here is added to the system.user collection of your repository database.

- 8 Enter a password name in the **Password** field.
You need to ensure that the password you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the password you enter here is added to the system.user collection of your repository database.
- 9 Enter a name in the **Repository name** field.
The name you specify here corresponds to the name used for the repository database and therefore cannot contain the following characters "/", "\", ".", "*", "<", ">", ":", "|", "?", or double quotes.
- 10 (Optional) You can unselect the **Save database user and password** check box.
 - This option is selected by default so that users can quickly and easily connect to the workspace, repository, or both by selecting a connection in the Manage Connections, Import, or Export wizards. The password is encrypted by default. See "[Saving User Name and Passwords](#)" on page 43.
 - If this option is unselected, user names and passwords are not persisted in the system, however it also means that users will be prompted whenever a connection to the workspace, repository, or both needs to be made.
- 11 Click **Details** to see the configuration parameters of the repository type you selected.
If you view the Details page and want to edit the configuration to add or delete a service, see "[Editing a Repository Configuration in the Blaze Advisor IDE](#)" on page 39.
- 12 Click **OK**.
If you need to create a connection to this repository, see "[Connecting to a MongoDB Repository Using the Blaze Advisor IDE](#)" on page 378.
After you have created the repository with a shared workspace, you may want to verify the versioning commands, see "[Verifying Blaze Advisor Versioning Commands](#)" on page 230.

Creating a MongoDB Repository (BVS versioning - private local file workspace) using the Blaze Advisor IDE

You can create a MongoDB Repository (BVS versioning - private local file workspace) if you want a repository with a versioning service which is similar to the other Blaze Advisor SCM repositories such as CVS, SVN, and ClearCase repositories in Blaze Advisor. In this case, the repository is hosted by the MongoDB database but the file workspace is stored on a file system. Because it uses local file workspaces, this repository configuration is not cloud compatible but can be converted to a cloud compatible configuration if necessary.

If you create this type of MongoDB repository, users need to connect to the repository hosted by MongoDB and their own private file workspace.

This repository configuration can be used with a MongoDB database instance that is configured in authenticated mode with a set of registered users. It can also be used with

a MongoDB database instance that is configured in non-authenticated mode in which case the only authentication is the one provided by the Blaze Advisor repository configured with the versioning service. See [“MongoDB Authentication and Blaze Advisor Repositories” on page 314](#).

After you create your repository, if you want to provide the connection information to other users, see [“Exporting a Blaze Advisor Workspace or Repository Connection” on page 62](#).

To create a MongoDB repository (BVS private local file workspace)

- 1 Select **Repository > New Repository**.
- 2 In the Create a Repository wizard, enter a name in the **Repository Nickname** field. This name is used to identify the repository connection in the IDE.
- 3 Select the **MongoDB Repository (BVS private local file workspace)** from the Repository Type dropdown.
- 4 Enter the **MongoDB URL** for the machine where the MongoDB database is running and enter a port number using the format <machineName or IP address>:<portNumber>. If you use the default MongoDB port number, entering it in the URL field is optional.

For example, 127.0.0.1:27017.

See [“MongoDB URL Connection Parameter” on page 315](#).
- 5 Select a **MongoDB authentication type** based on the authentication mode of the MongoDB database you want to use.
 - **None**
 - **MongoDB**

See [“MongoDB Authentication Type Connection Parameter” on page 315](#).
- 6 (Optional) Select **Use SSL (Requires MongoDB Enterprise Edition)**.

See [“SSL Connections and MongoDB Repositories” on page 383](#).
- 7 Enter a user name in the **User** field.

You need to ensure that the user name you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the user name you enter here is added to the system.user collection of your repository database.
- 8 Enter a password name in the **Password** field.

You need to ensure that the password you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the password you enter here is added to the system.user collection of your repository database.

9 Enter a name in the **Repository name** field.

The name you specify here corresponds to the name used for the repository database and therefore cannot contain the following characters "/", "\", ".", "*", "<", ">", ":"; "|"; "?", or double quotes.

10 (Optional) You can unselect the **Save database user and password** check box.

- This option is selected by default so that users can quickly and easily connect to the workspace, repository, or both by selecting a connection in the Manage Connections, Import, or Export wizards. The password is encrypted by default. See "[Saving User Name and Passwords](#)" on page 43.
- If this option is unselected, user names and passwords are not persisted in the system, however it also means that users will be prompted whenever a connection to the workspace, repository, or both needs to be made.

11 Enter or browse to a location for the file workspace in the **Local Folder** field.

12 Enter a user name in the **User** field.

You need to ensure that the user name you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the user name you enter here is added to the system.user collection of your repository database. You may be prompted for administrative credentials if necessary to accomplish this.

13 Enter a password name in the **Password** field.

You need to ensure that the password you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the password you enter here is added to the system.user collection of your repository database. You may be prompted for administrative credentials if necessary to accomplish this.

14 (Optional) You can select the **Save workspace user and password** check box to use a link to quickly and easily connect to the workspace or repository in Blaze Advisor. See "[Saving User Name and Passwords](#)" on page 43.

15 Click **Details** to see the configuration parameters of the repository type you selected.

- a** (Optional) You can allow distinct repository and workspace users, see "[Allowing Distinct Repository and Workspace Users](#)" on page 49.

- b** Click **OK**.

If you view the Details page and want to edit the configuration to add or delete a service, see "[Editing a Repository Configuration in the Blaze Advisor IDE](#)" on page 39.

16 Click OK.

If you need to create a workspace for the repository you just created, see “[Creating or Removing a MongoDB or File Workspace](#)” on page 365.

If you need to create a connection to this repository, see “[Connecting to a MongoDB Repository Using the Blaze Advisor IDE](#)” on page 378.

After you have created the repository with a private local file workspace, you may want to verify the versioning commands, see “[Verifying Blaze Advisor Versioning Commands](#)” on page 230.

Creating a MongoDB Repository Using the NdRomAdminUtil Utility

A repository is a storage mechanism for projects, folders, and entities. Repositories can have one or more services that provide additional functional features such as password encryption, versioning, or authorization. There are several default Blaze Advisor repository types that come pre-configured with service that are defined in the repository configuration. There are several different MongoDB repository configurations each with a set of default and specific services.

You can create a Blaze Advisor MongoDB repository that best suits your needs by connecting to a MongoDB instance. In order to connect to the MongoDB, you need to know its URL and whether or not it is running in authenticated or non-authenticated mode.



Important When you create a Blaze Advisor MongoDB repository or workspace, the connection information is cached by the MongoDB driver in the Blaze Advisor IDE or the RMA. To avoid an error, if for any reason, you need to stop and restart the MongoDB database, exit the Blaze Advisor IDE or the RMA prior to restarting MongoDB.

You can use the `NdRomAdminUtil` utility `createRepository` command to create a MongoDB repository using the command line. Prior to running the utility with the `createRepository` command, you need to prepare three files. You can think of these files as building blocks that provide instructions to Blaze Advisor about what type of repository you want to create and what services you want the repository to use:

- system connection

The system connection configuration contains tags that will allow you to connect to the Admin repository located in `<ADVISOR_HOME>/lib` and installed with Blaze Advisor. When you create a repository using the `NdRomAdminUtil` utility, you need to connect to the Admin repository so that the templates and instances and the system folder that control repository functionality are added to the new repository. When you create a repository using the Blaze Advisor IDE, a connection to the Admin repository occurs automatically.

- repository connection

The repository connection configuration file contains the tags for the specific type of repository you want to create. In this case, you need to add MongoDB specific tag names and values.

- rom configuration

The repository configuration file contains the details of the services you want your repository configuration to include. When you use the `NdRomAdminUtil` utility you need to explicitly add some of the specific services you want to use with your repository to the configuration file.

Most of these files can be reused when running the `NdRomAdminUtil` utility for other repository tasks. See ["The NdRomAdminUtil Utility Commands" on page 162](#).

You can create the following MongoDB Repository configurations using the `NdRomAdminUtil` utility:

- ["Creating a MongoDB Repository \(no versioning\) Using the NdRomAdminUtil Utility" on page 339](#)
- ["Creating a MongoDB Repository \(with Authorization manager\) Using the NdRomAdminUtil utility" on page 346](#)
- ["Creating a MongoDB Repository \(BVS versioning - private\) Using the NdRomAdminUtil utility" on page 350](#)
- ["Creating a MongoDB Repository \(BVS versioning - shared\) Using the NdRomAdminUtil utility" on page 355](#)
- ["Creating a MongoDB Repository \(BVS versioning - private local file workspace\) Using the NdRomAdminUtil utility" on page 360](#)

If you create a MongoDB Repository (versioning private) or MongoDB Repository (BVS private local file workspace) you need to create a workspace for each of your users. You can create a workspace using the Blaze Advisor IDE or the `NdRomAdminUtil` utility. See ["Creating or Removing a MongoDB or File Workspace" on page 365](#).

If a MongoDB workspace or a repository was created for you or you need to create a new connection to a MongoDB workspace or repository, see ["Connecting to a MongoDB Repository Using the Blaze Advisor IDE" on page 378](#).

Creating a MongoDB Repository (no versioning) Using the NdRomAdminUtil Utility

You can create a MongoDB repository (no versioning) for projects where security is not the foremost concern. This repository configuration is used with MongoDB running in non-authenticated mode (the default configuration of the MongoDB database).

After you create this repository, in the MongoDB database, any user who knows the URL and repository name can connect to the repository and create, update, and delete projects and project items.

You can create a MongoDB Repository (no versioning) using the `NdRomAdminUtil` utility by preparing the following files:

- “[System Connection Configuration File for the Admin Repository](#)” on page 340
- “[MongoDB Repository \(no versioning\) Connection Configuration File](#)” on page 341
- “[Repository Configuration File for a MongoDB Repository \(no versioning\)](#)” on page 343
- “[Creating a Batch File or Running NdRomAdminUtil Utility to Create a MongoDB Repository](#)” on page 344

System Connection Configuration File for the Admin Repository

You write an XML file (.cfg) for the Admin Repository when you want to use the `NdRomAdminUtil` utility to create a repository. The connection file for this repository contains information including the physical location of the Admin Repository in your Blaze Advisor installation, the repository name, and the user name. You write this file using a text editor.

The Admin Repository is a file type repository that is installed with Blaze Advisor. The Admin Repository contains the connection and configuration templates and instances for the repository types supported by Blaze Advisor. It also contains the system folder that contains the templates and instance files used for querying, filtering, verification, and management properties. When you create a repository using the Blaze Advisor IDE, a copy of this system folder and its contents are automatically added to your repository. When you create a repository using the utility, you need to reference the location of the Admin Repository and its system folder in a connection configuration file so that a copy of the system folder is added to the root directory of the repository.

After you have written the Admin Repository connection configuration file, you can use it anytime you want to use the `NdRomAdminUtil` utility where a reference to a Admin Repository connection file is required. For example, you can use this file for when creating another repository, creating a workspace or removing a repository or workspace.

The file should contain the following set of tags and values:



Note For reasons of brevity, a pair of element tags will be represented by the following notation, <example/>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<RepositoryConnection/>	The RepositoryConnection tag is used when you need to create a repository connection. All other tags in this table are nested within this tag.

Tag	Description
<Factory/>	The Factory tag is used when you want to create an instance of a class. The Admin Repository is a File repository. To create an instance of a connection for a File repository, you use the fully qualified class name: com.blazesoft.repository.file.NdFileRepositoryConnection.
<Name/>	The Name tag is used to specify the connection name listed in the Manage Connections wizard in Blaze Advisor. Because you can connect to the Admin Repository by enabling the Local Admin Repository command in the Repository menu instead of the Manage Connections wizard, you can use "Admin Repository" as the value within these tags.
<RepositoryFolder/>	The RepositoryFolder is used to specify the physical location of the Admin Repository in Blaze Advisor.
<User/>	The User tag is used to specify the user name. By default the user name for the Admin Repository is admin.
<RepositoryName/>	The RepositoryName tag is used to specify the name for the root directory. By default, the repository name for the Admin Repository is Admin Repository.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<RepositoryConnection>
    <Factory>
        com.blazesoft.repository.file.NdFileRepositoryConnection
    </Factory>
    <!--The RepositoryFolder tag contains the location of your Admin Repository in <ADVISOR_HOME>-->
    <Name>Admin Repository</Name>
    <RepositoryFolder>C:\Blaze\Advisor\lib\Admin Repository
    </RepositoryFolder>
    <User>admin </User>
    <RepositoryName> Admin Repository </RepositoryName>
</RepositoryConnection>
```

A sample Admin repository connection configuration file, `adminConfig.cfg` is located in `<ADVISOR_HOME>/lib`. Save a backup copy of this file if you decide to edit its contents.

MongoDB Repository (no versioning) Connection Configuration File

You write a repository connection configuration file with information including the connection class, the location of the repository directory, the repository connection

name, and the repository name. This is similar to the information you would enter in the New Repository wizard in the Blaze Advisor IDE.

You can create the MongoDB Repository (no versioning) connection configuration using a text editor and save the file with a .cfg extension. The connection configuration file contains the following tag structure and values:

 **Note** For reasons of brevity, a pair of element tags will be represented by the following notation, <example>/>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<RepositoryConnection/>	The RepositoryConnection tag is used when you need to create a repository connection. All other tags in this table are nested within this tag.
<Factory/>	The Factory tag is used when you want to create an instance of a class. To create an instance of a connection for a MongoDB non-versioned repository, you use the fully qualified class name: com.blazesoft.repository.mongodb.NdMongo RepositoryConnection.
<MongoDBAuthenticationType/>	Not used for non-versioned MongoDB repositories. By default, the value is 0 (for None or no authentication). See “ MongoDB Authentication Type Connection Parameter ” on page 315.
<MongoDBUrl/>	The URL of the MongoDB instance. Generally, when you create a MongoDB repository (no versioning) the URL is 127.0.0.1 and you are using the default MongoDB port, 27017. See “ MongoDB URL Connection Parameter ” on page 315. Example: 127.0.0.1:27017
<Name/>	The name is used to specify the connection name listed in the Manage Connections wizard in the Blaze Advisor IDE.
<RepositoryName/>	The name you specify here corresponds to the name used for the repository database and therefore cannot contain the following characters “/”, “\”, “.”, “*”, “<”, “>”, “:”, “ ”, “?”, or double quotes.
<User/>	The user name. There is no authentication for this repository configuration.
<Password/>	The password. There is no authentication for this repository configuration.
<SharedWorkspace/>	Value is false. This value is true only if you are creating a MongoDB workspace with a shared workspace.

Tag	Description
<UseSSLEnabled/>	Value is false. This value is true only if you are using a Secure Sockets Layer (SSL) available only with MongoDB Enterprise Edition. See “ SSL Connections and MongoDB Repositories ” on page 383.
<WorkspaceConnection/>	Value is false. This value is true only if this is a workspace connection.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RepositoryConnection>
    <Factory>com.blazesoft.repository.mongodb.NdMongo
    RepositoryConnection </Factory>
    <MongoDBAuthenticationType>1</MongoDBAuthenticationType>
    <MongoDBUrl> 127.0.0.1:27017 </MongoDBUrl>
    <Name>MongoDB1a</Name>
    <RepositoryName>MongoDB1a</RepositoryName>
    <User>admin</User>
    <Password>admin</Password>
    <SharedWorkspace>false</SharedWorkspace>
    <UseSSLEnabled>false</UseSSLEnabled>
    <WorkspaceConnection>false</WorkspaceConnection>
</RepositoryConnection>
```

A sample MongoDB repository connection configuration file, `mongoNVRep.cfg` is located in `<ADVISOR_HOME>/lib`. Save a backup copy of this file if you decide to edit its contents.

Repository Configuration File for a MongoDB Repository (no versioning)

You write a configuration file when you are creating your repository using the `NdRomAdminUtil` utility. This configuration file contains the class names for the services you want for your repository. What you add to this configuration file is similar to what you see by default in the Details page of the Create a Repository wizard in the Blaze Advisor IDE.

You can create the file using a text editor and save the file with a `.cfg` extension. The following is an example of a repository configuration file that can be used to create any non-versioned repository:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RomConfig>

    <RomQueryManagerConfig>
        <RomQueryManagerFactory>
            <JavaName>com.blazesoft.template.repository.query.NdRomDefault
            QueryManager</JavaName>
```

```
</RomQueryManagerFactory>
</RomQueryManagerConfig>
<RomConnectionManagerConfig>
    <ConnectionMode> 0 </ConnectionMode>
        <RomConnectionManagerFactory>
            <JavaName>com.blazesoftware.template.repository.impl.NdDefaultRom
                ConnectionManager</JavaName>
        </RomConnectionManagerFactory>
    </RomConnectionManagerConfig>

<RomSchemaManagerConfig>
    <RomSchemaManagerFactory>
        <JavaName>com.blazesoftware.template.repository.impl.NdDefaultRom
            SchemaManager</JavaName>
    </RomSchemaManagerFactory>
</RomSchemaManagerConfig>
<RepositoryConfig>

    <RepositoryEncryptionManagerConfig>

        <EncryptionCipherFactory>
            <DotNETCipher> TripleDES </DotNETCipher>
            <DotNETCipherParameterFactory>
                com.blazesoftware.util.crypto.dotnet.NdDotNETTripleDESCipher
                    ParameterFactory </DotNETCipherParameterFactory>
            <JavaCipher> TripleDES </JavaCipher>
            <JavaCipherParameterFactory>
                com.blazesoftware.util.crypto.java.NdJavaTripleDESKeySpecFactory
                    </JavaCipherParameterFactory>
        </EncryptionCipherFactory>

        <RepositoryEncryptionManagerFactory>
            <JavaName>
                com.blazesoftware.repository.base.NdDefaultRepository
                    EncryptionManager </JavaName>
            </RepositoryEncryptionManagerFactory>
        </RepositoryEncryptionManagerConfig>
    </RepositoryConfig>

    <AllowUnusedValues> true </AllowUnusedValues>
</RomConfig>
```

For information about services you can include in a repository configuration file, see [“Writing a Repository Configuration File” on page 191](#).

A sample repository configuration file `romConfig.cfg` is located in `<ADVISOR_HOME>/lib`. Save a backup copy of this file if you decide to edit its contents.

Creating a Batch File or Running NdRomAdminUtil Utility to Create a MongoDB Repository

You can either run the `NdRomAdminUtil` utility directly from the command line or you can write a batch file referencing the utility to create a repository.

- If you want to use the `NdRomAdminUtil` utility directly, you need to change the directory to `<ADVISOR_HOME>/bin` and enter the command, arguments and file names below.
- If you want to write a batch file, you enter the command, arguments and file names below to a file and then call the file from the command line.

Command, Argument, or Option	Description
<code>createRepository</code>	Command to create a repository
<code>-repositoryConnection</code>	Argument takes the path to the location of the repository connection configuration file (.cfg). This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
<code>-systemconnection</code>	Argument takes the path to the location of the system connection configuration file (.cfg). This file contains information such as the name of the repository which in this case is Admin Repository, the location of the Admin Repository and the user name.
<code>-configuration</code>	Argument takes the path to the location of the configuration file(.cfg).
<code>-verbose</code>	Option prints out all details messages into the command console.
<code>-m</code>	Option to add a comment when creating a repository.
<code>-debug</code>	Option prints out tracing information into the command console.

Example:

```
setenv
java com.blazesoft.template.repository.admin.NdRomAdminUtil
createRepository
-repositoryConnection "c:/Repo_Connections/ConnectionFiles/
RepoConnectionConfigFile.cfg"
-systemConnection "c:/Repo_Connections/ConnectionFiles/sysAdmin.cfg" -
-configuration "c:/Repo_Connections/ConnectionFiles/
romNonVersionedConfig.cfg" -m "MongoDB repo created" -verbose
```

You reference the path to these files as argument values for other tasks using the `NdRomAdminUtil` utility. See ["The NdRomAdminUtil Utility Commands" on page 162](#).

If you have created a MongoDB repository with BVS Versioning, After you have created the repository with a private MongoDB workspace a shared MongoDB workspace or a private local file workspace, you may want to verify the versioning commands, see ["Verifying Blaze Advisor Versioning Commands" on page 230](#).

Creating a MongoDB Repository (with Authorization manager) Using the NdRomAdminUtil utility

You can create a MongoDB repository with an authorization service and no versioning service if you want to control access to items in your projects using an authorization manager class.

This repository configuration can be used with a MongoDB database instance that is configured in authenticated mode with a set of registered users. It can also be used with a MongoDB database instance that is configured in non-authenticated mode in which case the only authentication is the one provided by the Blaze Advisor repository configured with an authorization manager. See “[MongoDB Authentication and Blaze Advisor Repositories](#)” on page 314.

You can create a MongoDB Repository (with Authorization manager) using the `NdRomAdminUtil` utility by preparing the following files:

- “[System Connection Configuration File for the Admin Repository](#)” on page 340
- “[MongoDB Repository \(with Authorization manager\) Connection Configuration File](#)” on page 346
- “[Repository Configuration File for a MongoDB Repository \(with Authorization manager\)](#)” on page 349
- “[Creating a Batch File or Running NdRomAdminUtil Utility to Create a MongoDB Repository](#)” on page 344

MongoDB Repository (with Authorization manager) Connection Configuration File

You write a repository connection configuration file with information including the connection class, the location of the repository directory, the repository connection name, and the repository name. This is similar to the information you would enter in the New Repository wizard in the Blaze Advisor IDE.

You can create the MongoDB Repository (no versioning) connection configuration using a text editor and save the file with a `.cfg` extension. The connection configuration file contains the following tag structure and values:



Note For reasons of brevity, a pair of element tags will be represented by the following notation, `<example/>`.

Tag	Description
<code><?xml version="1.0" encoding="UTF-8" ?></code>	Processing information that contains the XML version and character-encoding used.
<code><RepositoryConnection/></code>	The <code>RepositoryConnection</code> tag is used when you need to create a repository connection. All other tags in this table are nested within this tag.

Tag	Description
<Factory/>	The Factory tag is used when you want to create an instance of a class. To create an instance of a connection for a MongoDB non-versioned repository, you use the fully qualified class name: com.blazesoft.repository.mongodb.NdMongo RepositoryConnection.
<MongoDBAuthenticationType/>	The authentication type of the MongoDB instance. Available values are: ■ 0 -- None ■ 1 -- MongoDB See " MongoDB Authentication Type Connection Parameter " on page 315.
<MongoDBUrl/>	The URL of the MongoDB instance. See " MongoDB URL Connection Parameter " on page 315. Example: 127.0.0.1:27017
<Name/>	The name is used to specify the connection name listed in the Manage Connections wizard in the Blaze Advisor IDE.
<RepositoryName/>	The name you specify here corresponds to the name used for the repository database and therefore cannot contain the following characters "/", "\", ".", "*", "<", ">", ":", " ", "?", or double quotes.
<User/>	The user name. ■ If the MongoDB database you are connecting to is running in authenticated mode and you have connected using the MongoDB authentication type, the user name is added to the system.user collection and is used by MongoDB to authenticate the user. The authorization manager also uses the same user name to authenticate the user for access to items in the repository or workspace. ■ If the MongoDB database you are connecting to is running in unauthenticated mode and you have connected using the None authentication type, the MongoDB does not authenticate the user. The authorization manager uses the user name to authenticate the user for access to items in the repository or workspace.

Tag	Description
<Password/>	<p>The password.</p> <ul style="list-style-type: none"> ■ If the MongoDB database you are connecting to is running in authenticated mode and you have connected using the MongoDB authentication type, the password is added to the system.user collection and is used by MongoDB to authenticate the user. The authorization manager also uses the same password to authenticate the user for access to items in the repository or workspace. ■ If the MongoDB database you are connecting to is running in unauthenticated mode and you have connected using the None authentication type, the MongoDB does not authenticate the user. The authorization manager uses the password to authenticate the user for access to items in the repository or workspace.
<SharedWorkspace/>	Value is false. This value is true only if you are creating a MongoDB workspace with a shared workspace.
<UseSSLEnabled/>	Value is false. This value is true only if you are using a Secure Sockets Layer (SSL) available only with MongoDB Enterprise Edition. See “ SSL Connections and MongoDB Repositories ” on page 383.
<WorkspaceConnection/>	Value is false. This value is true only if this is a workspace connection.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RepositoryConnection>
    <Factory>com.blazesoft.repository.mongodb.NdMongo
        RepositoryConnection </Factory>
    <MongoDBAuthenticationType>1</MongoDBAuthenticationType>
    <MongoDBUrl> 127.0.0.1:27017 </MongoDBUrl>
    <Name>MongoDB1a</Name>
    <RepositoryName>MongoDB1a</RepositoryName>
    <User>admin</User>
    <Password>admin</Password>
    <SharedWorkspace>false</SharedWorkspace>
    <UseSSLEnabled>false</UseSSLEnabled>
    <WorkspaceConnection>false</WorkspaceConnection>
</RepositoryConnection>
```

A sample MongoDB repository configuration file `mongoNVRep.cfg` is located in `<ADVISOR_HOME>/lib`. Save a backup copy of this file if you decide to edit its contents.

Repository Configuration File for a MongoDB Repository (with Authorization manager)

You write a configuration file when you are creating your repository using the `NdRomAdminUtil` utility. This configuration file contains the class names for the services you want for your repository. What you add to this configuration file is similar to what you see by default in the Details page of the Create a Repository wizard in the Blaze Advisor IDE. To add an authorization service to a MongoDB Repository (with Authorization manager), you need to add the fully qualified name of the custom Authorization Manager class to your repository configuration.

You can create the file using a text editor and save the file with a `.cfg` extension. The following is an example of a repository configuration file that can be used to create any non-versioned repository:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RomConfig>

    <RomQueryManagerConfig>
        <RomQueryManagerFactory>
            <JavaName>com.blazesoftware.template.repository.query.NdRomDefault
                QueryManager</JavaName>
        </RomQueryManagerFactory>
    </RomQueryManagerConfig>
    <RomConnectionManagerConfig>
        <ConnectionMode> 0 </ConnectionMode>
        <RomConnectionManagerFactory>
            <JavaName>com.blazesoftware.template.repository.impl.NdDefaultRom
                ConnectionManager</JavaName>
        </RomConnectionManagerFactory>
    </RomConnectionManagerConfig>

    <RomSchemaManagerConfig>
        <RomSchemaManagerFactory>
            <JavaName>com.blazesoftware.template.repository.impl.NdDefaultRom
                SchemaManager</JavaName>
        </RomSchemaManagerFactory>
    </RomSchemaManagerConfig>
    <RepositoryConfig>
        <ItemContentCompressionThresholdKB> -1
            </ItemContentCompressionThresholdKB>

        <RepositoryAuthorizationManagerFactory>
            <JavaName>com.examples.ExampleAuthorizationManager</JavaName>
        </RepositoryAuthorizationManagerFactory>

        <RepositoryEncryptionManagerConfig>

            <EncryptionCipherFactory>
                <DotNETCipher> TripleDES </DotNETCipher>
                <DotNETCipherParameterFactory>
                    com.blazesoftware.util.crypto.dotnet.NdDotNETTripleDESCipher
                        ParameterFactory </DotNETCipherParameterFactory>
                <JavaCipher> TripleDES </JavaCipher>
            </EncryptionCipherFactory>
        </RepositoryEncryptionManagerConfig>
    </RepositoryConfig>
</RomConfig>
```

```
<JavaCipherParameterFactory>
    com.blazesoft.util.crypto.java.NdJavaTripleDESKeySpecFactory
        </JavaCipherParameterFactory>
</EncryptionCipherFactory>

<RepositoryEncryptionManagerFactory>
<JavaName>
    com.blazesoft.repository.base.NdDefaultRepository
        EncryptionManager </JavaName>
</RepositoryEncryptionManagerFactory>

</RepositoryEncryptionManagerConfig>

</RepositoryConfig>

<AllowUnusedValues> true </AllowUnusedValues>
</RomConfig>
```

For information about services you can include in a repository configuration file, see [“Writing a Repository Configuration File” on page 191](#).

A sample MongoDB repository configuration file `romConfig.cfg` is located in `<ADVISOR_HOME>/lib`. You need to edit this file to include an authorization manager class as shown above. Save a backup copy of this file if you decide to edit its contents.

Creating a MongoDB Repository (BVS versioning - private) Using the NdRomAdminUtil utility

You can create a MongoDB Repository (BVS versioning - private) if you want to use a versioning service with your cloud compatible repository with no local file IO. In this case, the repository and the private workspaces for each user are hosted by the MongoDB database.

This repository configuration can be used with a MongoDB database instance that is configured in authenticated mode with a set of registered users. It can also be used with a MongoDB database instance that is configured in non-authenticated mode in which case the only authentication is the one provided by the Blaze Advisor repository configured with the versioning service. See [“MongoDB Authentication and Blaze Advisor Repositories” on page 314](#).

You can create a MongoDB Repository (versioning - private) using the `NdRomAdminUtil` utility by preparing the following files:

- [“System Connection Configuration File for the Admin Repository” on page 340](#)
- [“MongoDB Repository \(versioning - private\) Connection Configuration File” on page 351](#)
- [“Repository Configuration File for a MongoDB Repository \(versioning - private\)” on page 353](#)
- [“Creating a Batch File or Running NdRomAdminUtil Utility to Create a MongoDB Repository” on page 344](#)

After you create the repository, you need to create workspaces for each user.

- [“Creating a Workspace for a MongoDB Repository \(versioning - private\) Using the NdRomAdminUtil Utility” on page 370.](#)
- You can also create your private MongoDB workspaces using the IDE, see [“Creating a Workspace for a MongoDB Repository \(versioning - private\) Using the Blaze Advisor IDE” on page 365.](#)

MongoDB Repository (versioning - private) Connection Configuration File

You write a repository connection configuration file with information including the connection class, the location of the repository directory, the repository connection name, and the repository name. This is similar to the information you would enter in the New Repository wizard in the Blaze Advisor IDE.

You can create the MongoDB Repository (versioning - private) connection configuration using a text editor and save the file with a .cfg extension. The connection configuration file contains the following tag structure and values:



Note For reasons of brevity, a pair of element tags will be represented by the following notation, <example>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<VersioningRepository Connection/>	The VersioningRepositoryConnection tag is used when you need to create a connection for a versioned repository. All other tags in this table are nested within this tag.
<PersistCredentials/>	Whether or not the credentials of the user are persisted in the repository configuration files. Available values are: <ul style="list-style-type: none"> ■ true ■ false See “Saving User Name and Passwords” on page 43.
<RepositoryConnection/>	The RepositoryConnection tag is used when you need to create a repository connection. All other connection tags in this table are nested within this tag.
<Factory/>	The Factory tag is used when you want to create an instance of a class. To create an instance of a connection for a MongoDB non-versioned repository, you use the fully qualified class name: com.blazesoftware.repository.mongodb.NdMongo RepositoryConnection.

Tag	Description
<MongoDBAuthenticationType/>	<p>The authentication type of the MongoDB instance. Available values are:</p> <ul style="list-style-type: none"> ■ 0 -- None ■ 1 -- MongoDB <p>See “MongoDB Authentication Type Connection Parameter” on page 315.</p>
<MongoDBUrl/>	<p>The URL of the MongoDB instance. See “MongoDB URL Connection Parameter” on page 315.</p> <p>Example: 127.0.0.1:27017</p>
<Name/>	<p>The name is used to specify the connection name listed in the Manage Connections wizard in the Blaze Advisor IDE.</p>
<RepositoryName/>	<p>The name you specify here corresponds to the name used for the repository database and therefore cannot contain the following characters “/”, “\”, “.”, “*”, “<”, “>”, “:”, “ ”, “?”, or double quotes.</p>
<User/>	<p>The user name.</p> <ul style="list-style-type: none"> ■ If the MongoDB database you are connecting to is running in authenticated mode and you have connected using the MongoDB authentication type, the user name is added to the system.user collection and is used by MongoDB to authenticate the user. The authorization manager also uses the same user name to authenticate the user for access to items in the repository or workspace. ■ If the MongoDB database you are connecting to is running in unauthenticated mode and you have connected using the None authentication type, the MongoDB does not authenticate the user. The authorization manager uses the user name to authenticate the user for access to items in the repository or workspace.
<Password/>	<p>The password.</p> <ul style="list-style-type: none"> ■ If the MongoDB database you are connecting to is running in authenticated mode and you have connected using the MongoDB authentication type, the password is added to the system.user collection and is used by MongoDB to authenticate the user. The authorization manager also uses the same password to authenticate the user for access to items in the repository or workspace. ■ If the MongoDB database you are connecting to is running in unauthenticated mode and you have connected using the None authentication type, the MongoDB does not authenticate the user. The authorization manager uses the password to authenticate the user for access to items in the repository or workspace.

Tag	Description
<SharedWorkspace/>	Value is false. This value is true only if you are creating a MongoDB workspace with a shared workspace.
<UseSSLEnabled/>	Value is false. This value is true only if you are using a Secure Sockets Layer (SSL) available only with MongoDB Enterprise Edition. See “ SSL Connections and MongoDB Repositories ” on page 383.
<WorkspaceConnection/>	Value is false. This value is true only if this is a workspace connection.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<VersioningRepositoryConnection>
    <Factory>com.blazesoft.repository.generic.version.NdNativeVersioning
        RepositoryConnection </Factory>
    <PersistCredentials> true </PersistCredentials>
    <RepositoryConnection>
        <Factory>com.blazesoft.repository.mongodb.NdMongo
            RepositoryConnection </Factory>
        <MongoDBAuthenticationType>1</MongoDBAuthenticationType>
        <MongoDBUrl> 127.0.0.1:27017 </MongoDBUrl>
        <Name>MongoDB1a</Name>
        <RepositoryName>MongoDB1a</RepositoryName>
        <User>admin</User>
        <Password>admin</Password>
        <SharedWorkspace>false</SharedWorkspace>
        <UseSSLEnabled>false</UseSSLEnabled>
        <WorkspaceConnection>false</WorkspaceConnection>
    </RepositoryConnection>
</VersioningRepositoryConnection>
```

A sample MongoDB repository connection configuration file `mongoBVSRepository.cfg` is located in `<ADVISOR_HOME>/lib`. Save a backup copy of this file if you decide to edit its contents.

Repository Configuration File for a MongoDB Repository (versioning - private)

You write a configuration file when you are creating your repository using the `NdRomAdminUtil` utility. This configuration file contains the class names for the services you want for your repository. What you add to this configuration file is similar to what you see by default in the Details page of the Create a Repository wizard in the Blaze Advisor IDE. To add a Blaze versioning service to a MongoDB Repository (versioning - private), you need to add Version Manager configuration to your repository configuration.

You can create the file using a text editor and save the file with a .cfg extension. The following is an example of a repository configuration file that can be used to create any versioned repository:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RomConfig>
<AllowUnusedValues> true </AllowUnusedValues>
    <!--<AtticRepository> true </AtticRepository>-->
<PromProjectReleaseManagerConfig>

    <PromProjectReleaseManagerFactory>
        <JavaName> com.blazesoft.template.repository.impl.
            NdDefaultPromProjectReleaseManager </JavaName>
    </PromProjectReleaseManagerFactory>
</PromProjectReleaseManagerConfig>

<RomQueryManagerConfig>
    <RomQueryManagerFactory>
        <JavaName> com.blazesoft.template.repository.query.NdRom
            DefaultQueryManager </JavaName>
    </RomQueryManagerFactory>
</RomQueryManagerConfig>

<RomConnectionManagerConfig>
<ConnectionMode> 0 </ConnectionMode>
    <RomConnectionManagerFactory>
        <JavaName> com.blazesoft.template.repository.impl.NdDefault
            RomConnectionManager </JavaName>
    </RomConnectionManagerFactory>
</RomConnectionManagerConfig>

</RomConnectionManagerConfig>

<RomSchemaManagerConfig>
    <RomSchemaManagerFactory>
        <JavaName> com.blazesoft.template.repository.impl.NdDefault
            RomSchemaManager </JavaName>
    </RomSchemaManagerFactory>
</RomSchemaManagerConfig>

</RomSchemaManagerConfig>

<RepositoryConfig>
<ItemContentCompressionThresholdKB> -1 </ItemContentCompressionThresholdKB>

<RepositoryEncryptionManagerConfig>

    <EncryptionCipherFactory>
        <DotNETCipher> TripleDES </DotNETCipher>
            <DotNETCipherParameterFactory>
                com.blazesoft.util.crypto.dotnet.NdDotNETTriple
                    DESCipherParameterFactory </DotNETCipherParameterFactory>
        <JavaCipher> TripleDES </JavaCipher>
            <JavaCipherParameterFactory> com.blazesoft.util.crypto.java.
                NdJavaTripleDESKeySpecFactory </JavaCipherParameterFactory>
    </EncryptionCipherFactory>

    <RepositoryEncryptionManagerFactory>
```

```

<JavaName> com.blazesoft.repository.base.NdDefaultRepository
    EncryptionManager </JavaName>
</RepositoryEncryptionManagerFactory>
</RepositoryEncryptionManagerConfig>
    <AtticRepository> false </AtticRepository>
<RepositoryVersionManagerConfig>
    <Impersonate> false </Impersonate>
    <PrivateWorkspaceSupported> true </PrivateWorkspaceSupported>

    <RepositoryVersionManagerFactory>
        <JavaName> com.blazesoft.repository.generic.version.NdNative
            RepositoryVersionManager </JavaName>
        </RepositoryVersionManagerFactory>

    </RepositoryVersionManagerConfig>
</RepositoryConfig>

<AllowUnusedValues> true </AllowUnusedValues>

</RomConfig>

```

For information about services you can include in a repository configuration file, see [“Writing a Repository Configuration File” on page 191](#).

A sample MongoDB repository configuration file `romConfig.cfg` is located in `<ADVISOR_HOME>/lib`. Save a backup copy of this file if you decide to edit its contents.

Creating a MongoDB Repository (BVS versioning - shared) Using the NdRomAdminUtil utility

You can create a MongoDB Repository (BVS versioning - shared) if you want to use a versioning service with your cloud compatible repository. In this case, the repository and the shared workspace used by all users are hosted by the MongoDB database.

This repository configuration can be used with a MongoDB database instance that is configured in authenticated mode with a set of registered users. It can also be used with a MongoDB database instance that is configured in non-authenticated mode in which case the only authentication is the one provided by the Blaze Advisor repository configured with the versioning service. See [“MongoDB Authentication and Blaze Advisor Repositories” on page 314](#).

You can create a MongoDB Repository (versioning - private) using the `NdRomAdminUtil` utility by preparing the following files:

- [“System Connection Configuration File for the Admin Repository” on page 340](#)
- [“MongoDB Repository \(versioning - shared\) Connection Configuration File” on page 356](#)
- [“Repository Configuration File for a MongoDB Repository \(versioning - shared\)” on page 358](#)
- [“Creating a Batch File or Running NdRomAdminUtil Utility to Create a MongoDB Repository” on page 344](#)

MongoDB Repository (versioning - shared) Connection Configuration File

You write a repository connection configuration file with information including the connection class, the location of the repository directory, the repository connection name, and the repository name. This is similar to the information you would enter in the New Repository wizard in the Blaze Advisor IDE.

You can create the MongoDB Repository (versioning - shared) connection configuration using a text editor and save the file with a .cfg extension. The connection configuration file contains the following tag structure and values:

 **Note** For reasons of brevity, a pair of element tags will be represented by the following notation, <example/>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<VersioningRepositoryConnection/>	The VersioningRepositoryConnection tag is used when you need to create a connection for a versioned repository. All other tags in this table are nested within this tag.
<PersistCredentials/>	Whether or not the credentials of the user are persisted in the repository configuration files. Available values are: <ul style="list-style-type: none">■ true■ false See “Saving User Name and Passwords” on page 43.
<RepositoryConnection/>	The RepositoryConnection tag is used when you need to create a repository connection. All other connection tags in this table are nested within this tag.
<Factory/>	The Factory tag is used when you want to create an instance of a class. To create an instance of a connection for a MongoDB non-versioned repository, you use the fully qualified class name: com.blazesoft.repository.mongodb.NdMongoRepositoryConnection.
<MongoDBAuthenticationType/>	The authentication type of the MongoDB instance. Available values are: <ul style="list-style-type: none">■ 0 -- None■ 1 -- MongoDB See “MongoDB Authentication Type Connection Parameter” on page 315.
<MongoDBUrl/>	The URL of the MongoDB instance. See “MongoDB URL Connection Parameter” on page 315. Example: 127.0.0.1:27017

Tag	Description
<Name/>	The name is used to specify the connection name listed in the Manage Connections wizard in the Blaze Advisor IDE.
<RepositoryName/>	The name you specify here corresponds to the name used for the repository database and therefore cannot contain the following characters “/”, “\”, “.”, “*”, “<”, “>”, “:”, “ ”, “?”, or double quotes.
<User/>	<p>The user name.</p> <ul style="list-style-type: none"> ■ If the MongoDB database you are connecting to is running in authenticated mode and you have connected using the MongoDB authentication type, the user name is added to the system.user collection and is used by MongoDB to authenticate the user. The authorization manager also uses the same user name to authenticate the user for access to items in the repository or workspace. ■ If the MongoDB database you are connecting to is running in unauthenticated mode and you have connected using the None authentication type, the MongoDB does not authenticate the user. The authorization manager uses the user name to authenticate the user for access to items in the repository or workspace.
<Password/>	<p>The password.</p> <ul style="list-style-type: none"> ■ If the MongoDB database you are connecting to is running in authenticated mode and you have connected using the MongoDB authentication type, the password is added to the system.user collection and is used by MongoDB to authenticate the user. The authorization manager also uses the same password to authenticate the user for access to items in the repository or workspace. ■ If the MongoDB database you are connecting to is running in unauthenticated mode and you have connected using the None authentication type, the MongoDB does not authenticate the user. The authorization manager uses the password to authenticate the user for access to items in the repository or workspace.
<SharedWorkspace/>	Value is true because you are creating a MongoDB Repository (versioning - shared).
<UseSSLEnabled/>	Value is false. This value is true only if you are using a Secure Sockets Layer (SSL) available only with MongoDB Enterprise Edition. See “ SSL Connections and MongoDB Repositories ” on page 383.
<WorkspaceConnection/>	Value is false. This value is true only if this is a workspace connection.

 **Note** To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<VersioningRepositoryConnection>
    <Factory>com.blazesoft.repository.generic.version.NdNativeVersioning
        RepositoryConnection </Factory>
    <PersistCredentials> true </PersistCredentials>
    <RepositoryConnection>
        <Factory>com.blazesoft.repository.mongodb.NdMongo
            RepositoryConnection </Factory>
        <MongoDBAuthenticationType>1</MongoDBAuthenticationType>
        <MongoDBUrl> 127.0.0.1:27017 </MongoDBUrl>
        <Name>MongoDB1a</Name>
        <RepositoryName>MongoDB1a</RepositoryName>
        <User>admin</User>
        <Password>admin</Password>
        <SharedWorkspace>true</SharedWorkspace>
        <UseSSLEnabled>false</UseSSLEnabled>
        <WorkspaceConnection>false</WorkspaceConnection>
    </RepositoryConnection>
</VersioningRepositoryConnection>
```

A sample MongoDB repository connection configuration file `mongoBVSRepository.cfg` is located in `<ADVISOR_HOME>/lib`. Save a backup copy of this file if you decide to edit its contents.

Repository Configuration File for a MongoDB Repository (versioning - shared)

You write a configuration file when you are creating your repository using the `NdRomAdminUtil` utility. This configuration file contains the class names for the services you want for your repository. What you add to this configuration file is similar to what you see by default in the Details page of the Create a Repository wizard in the Blaze Advisor IDE. To add a Blaze versioning service to a MongoDB Repository (versioning - shared), you need to add Version Manager configuration to your repository configuration.

You can create the file using a text editor and save the file with a `.cfg` extension. The following is an example of a repository configuration file that can be used to create any versioned repository:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RomConfig>
<AllowUnusedValues> true </AllowUnusedValues>
    <AtticRepository> true </AtticRepository>
<PromProjectReleaseManagerConfig>

    <PromProjectReleaseManagerFactory>
        <JavaName> com.blazesoft.template.repository.impl.
            NdDefaultPromProjectReleaseManager </JavaName>
    </PromProjectReleaseManagerFactory>
</PromProjectReleaseManagerConfig>
```

```

<RomQueryManagerConfig>
    <RomQueryManagerFactory>
        <JavaName> com.blazesoft.template.repository.query.NdRom
        DefaultQueryManager </JavaName>
    </RomQueryManagerFactory>
</RomQueryManagerConfig>

<RomConnectionManagerConfig>
    <ConnectionMode> 0 </ConnectionMode>
    <RomConnectionManagerFactory>
        <JavaName> com.blazesoft.template.repository.impl.NdDefault
        RomConnectionManager </JavaName>
    </RomConnectionManagerFactory>
</RomConnectionManagerConfig>

<RomSchemaManagerConfig>
    <RomSchemaManagerFactory>
        <JavaName> com.blazesoft.template.repository.impl.NdDefault
        RomSchemaManager </JavaName>
    </RomSchemaManagerFactory>
</RomSchemaManagerConfig>

<RepositoryConfig>
    <ItemContentCompressionThresholdKB> -1 </ItemContentCompressionThresholdKB>

    <RepositoryEncryptionManagerConfig>

        <EncryptionCipherFactory>
            <DotNETCipher> TripleDES </DotNETCipher>
            <DotNETCipherParameterFactory>
                com.blazesoft.util.crypto.dotnet.NdDotNETTriple
                DESCipherParameterFactory </DotNETCipherParameterFactory>
            <JavaCipher> TripleDES </JavaCipher>
            <JavaCipherParameterFactory> com.blazesoft.util.crypto.java.
            NdJavaTripleDESKeySpecFactory </JavaCipherParameterFactory>
        </EncryptionCipherFactory>

        <RepositoryEncryptionManagerFactory>
            <JavaName> com.blazesoft.repository.base.NdDefaultRepository
            EncryptionManager </JavaName>
        </RepositoryEncryptionManagerFactory>
    </RepositoryEncryptionManagerConfig>
    <AtticRepository> false </AtticRepository>
    <RepositoryVersionManagerConfig>
        <Impersonate> false </Impersonate>
        <PrivateWorkspaceSupported> false </PrivateWorkspaceSupported>

        <RepositoryVersionManagerFactory>
            <JavaName> com.blazesoft.repository.generic.version.NdNative
            RepositoryVersionManager </JavaName>
        </RepositoryVersionManagerFactory>
    </RepositoryVersionManagerConfig>

```

```
</RepositoryConfig>  
  
<AllowUnusedValues> true </AllowUnusedValues>  
  
</RomConfig>
```

For information about services you can include in a repository configuration file, see ["Writing a Repository Configuration File" on page 191](#).

A sample MongoDB repository configuration file `romConfig.cfg` is located in `<ADVISOR_HOME>/lib`. Save a backup copy of this file if you decide to edit its contents.

Creating a MongoDB Repository (BVS versioning - private local file workspace) Using the NdRomAdminUtil utility

You can create a MongoDB Repository (BVS versioning - private local file workspace) if you want a repository with a versioning service which is similar to the other Blaze Advisor SCM repositories such as CVS, SVN, and ClearCase repositories in Blaze Advisor. In this case, the repository is hosted by the MongoDB database but the file workspace is stored on a file system. Because it uses local file workspaces, this repository configuration is not cloud compatible but can be converted to a cloud compatible configuration if necessary.

If you create this type of MongoDB repository, users need to connect to the repository hosted by MongoDB and their own private file workspace.

This repository configuration can be used with a MongoDB database instance that is configured in authenticated mode with a set of registered users. It can also be used with a MongoDB database instance that is configured in non-authenticated mode in which case the only authentication is the one provided by the Blaze Advisor repository configured with the versioning service. See ["MongoDB Authentication and Blaze Advisor Repositories" on page 314](#).

You can create a MongoDB Repository (BVS versioning - private local file) using the `NdRomAdminUtil` utility by preparing the following files:

- ["System Connection Configuration File for the Admin Repository" on page 340](#)
- ["MongoDB Repository \(BVS versioning - private local file workspace\) Connection Configuration File" on page 361](#)
- ["Repository Configuration File for a MongoDB Repository \(BVS versioning - private local file\)" on page 363](#)
- ["Creating a Batch File or Running NdRomAdminUtil Utility to Create a MongoDB Repository" on page 344](#)

After you create the repository, you need to create workspaces for each user.

- ["Creating a Workspace for a MongoDB Repository \(BVS versioning - private local file\) Using the NdRomAdminUtil Utility" on page 373.](#)
- You can also create your private local file workspaces using the IDE, see ["Creating a Workspace for a MongoDB Repository \(BVS versioning - private local file\) Using the Blaze Advisor IDE" on page 368.](#)

MongoDB Repository (BVS versioning - private local file workspace) Connection Configuration File

You write a repository connection configuration file with information including the connection class, the location of the repository directory, the repository connection name, and the repository name. This is similar to the information you would enter in the New Repository wizard in the Blaze Advisor IDE.

You can create the MongoDB Repository (BVS versioning - private local file workspace) connection configuration using a text editor and save the file with a .cfg extension. The connection configuration file contains the following tag structure and values:



Note For reasons of brevity, a pair of element tags will be represented by the following notation, <example/>.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<VersioningRepositoryConnection/>	The VersioningRepositoryConnection tag is used when you need to create a connection for a versioned repository. All other tags in this table are nested within this tag.
<PersistCredentials/>	Whether or not the credentials of the user are persisted in the repository configuration files. Available values are: <ul style="list-style-type: none"> ■ true ■ false See “ Saving User Name and Passwords ” on page 43.
<RepositoryConnection/>	The RepositoryConnection tag is used when you need to create a repository connection. All other connection tags in this table are nested within this tag.
<Factory/>	The Factory tag is used when you want to create an instance of a class. To create an instance of a connection for a MongoDB non-versioned repository, you use the fully qualified class name: com.blazesoftware.repository.mongodb.NdMongoRepositoryConnection.
<MongoDBAuthenticationType/>	The authentication type of the MongoDB instance. Available values are: <ul style="list-style-type: none"> ■ 0 -- None ■ 1 -- MongoDB See “ MongoDB Authentication Type Connection Parameter ” on page 315.
<MongoDBUrl/>	The URL of the MongoDB instance. See “ MongoDB URL Connection Parameter ” on page 315. Example: 127.0.0.1:27017

Tag	Description
<Name/>	The name is used to specify the connection name listed in the Manage Connections wizard in the Blaze Advisor IDE.
<RepositoryName/>	The name you specify here corresponds to the name used for the repository database and therefore cannot contain the following characters "/", "\", ".", "*", "<", ">", ":" , " ", "?", or double quotes.
<User/>	<p>The user name.</p> <ul style="list-style-type: none"> ■ If the MongoDB database you are connecting to is running in authenticated mode and you have connected using the MongoDB authentication type, the user name is added to the system.user collection and is used by MongoDB to authenticate the user. The authorization manager also uses the same user name to authenticate the user for access to items in the repository or workspace. ■ If the MongoDB database you are connecting to is running in unauthenticated mode and you have connected using the None authentication type, the MongoDB does not authenticate the user. The authorization manager uses the user name to authenticate the user for access to items in the repository or workspace.
<Password/>	<p>The password.</p> <ul style="list-style-type: none"> ■ If the MongoDB database you are connecting to is running in authenticated mode and you have connected using the MongoDB authentication type, the password is added to the system.user collection and is used by MongoDB to authenticate the user. The authorization manager also uses the same password to authenticate the user for access to items in the repository or workspace. ■ If the MongoDB database you are connecting to is running in unauthenticated mode and you have connected using the None authentication type, the MongoDB does not authenticate the user. The authorization manager uses the password to authenticate the user for access to items in the repository or workspace.
<SharedWorkspace/>	Value is false. This value is true only if you are creating a MongoDB workspace with a shared workspace.
<UseSSLEnabled/>	Value is false. This value is true only if you are using a Secure Sockets Layer (SSL) available only with MongoDB Enterprise Edition. See " SSL Connections and MongoDB Repositories " on page 383.
<WorkspaceConnection/>	Value is false. This value is true only if this is a workspace connection.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<VersioningRepositoryConnection>
    <Factory>com.blazesoft.repository.generic.version.NdNativeVersioning
        RepositoryConnection </Factory>
    <PersistCredentials> true </PersistCredentials>
    <RepositoryConnection>
        <Factory>com.blazesoft.repository.mongodb.NdMongo
            RepositoryConnection </Factory>
        <MongoDBAuthenticationType>1</MongoDBAuthenticationType>
        <MongoDBUrl> 127.0.0.1:27017 </MongoDBUrl>
        <Name>MongoDB1a</Name>
        <RepositoryName>MongoDB1a</RepositoryName>
        <User>admin</User>
        <Password>admin</Password>
        <SharedWorkspace>false</SharedWorkspace>
        <UseSSLEnabled>false</UseSSLEnabled>
        <WorkspaceConnection>false</WorkspaceConnection>
    </RepositoryConnection>
</VersioningRepositoryConnection>
```

A sample MongoDB repository connection configuration file `mongoBVSRepository.cfg` is located in `<ADVISOR_HOME>/lib`. Save a backup copy of this file if you decide to edit its contents.

Repository Configuration File for a MongoDB Repository (BVS versioning - private local file)

You write a configuration file when you are creating your repository using the `NdRomAdminUtil` utility. This configuration file contains the class names for the services you want for your repository. What you add to this configuration file is similar to what you see by default in the Details page of the Create a Repository wizard in the Blaze Advisor IDE. To add a Blaze versioning service to a MongoDB Repository (BVS versioning - private local file), you need to add Version Manager configuration to your repository configuration.

You can create the file using a text editor and save the file with a `.cfg` extension. The following is an example of a repository configuration file that can be used to create any versioned repository:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RomConfig>

    <RomQueryManagerConfig>
        <RomQueryManagerFactory>
            <JavaName>com.blazesoft.template.repository.query.NdRomDefault
                QueryManager</JavaName>
        </RomQueryManagerFactory>
    </RomQueryManagerConfig>
```

```
<RomConnectionManagerConfig>
    <ConnectionMode> 0 </ConnectionMode>
        <RomConnectionManagerFactory>
            <JavaName>com.blazesoftware.template.repository.impl.NdDefaultRom
                ConnectionManager</JavaName>
        </RomConnectionManagerFactory>
    </RomConnectionManagerConfig>

<RomSchemaManagerConfig>
    <RomSchemaManagerFactory>
        <JavaName>com.blazesoftware.template.repository.impl.NdDefaultRom
            SchemaManager</JavaName>
    </RomSchemaManagerFactory>
</RomSchemaManagerConfig>
<RepositoryConfig>
    <ItemContentCompressionThresholdKB> -1
        </ItemContentCompressionThresholdKB>

    <RepositoryEncryptionManagerConfig>
        <EncryptionCipherFactory>
            <DotNETCipher> TripleDES </DotNETCipher>
            <DotNETCipherParameterFactory>
                com.blazesoftware.util.crypto.dotnet.NdDotNETTripleDESCipher
                    ParameterFactory </DotNETCipherParameterFactory>
            <JavaCipher> TripleDES </JavaCipher>
            <JavaCipherParameterFactory>
                com.blazesoftware.util.crypto.java.NdJavaTripleDESKeySpecFactory
                    </JavaCipherParameterFactory>
        </EncryptionCipherFactory>

        <RepositoryEncryptionManagerFactory>
            <JavaName>
                com.blazesoftware.repository.base.NdDefaultRepository
                    EncryptionManager </JavaName>
            </RepositoryEncryptionManagerFactory>
        </RepositoryEncryptionManagerConfig>
        <AtticRepository> false </AtticRepository>

        <RepositoryVersionManagerConfig>
            <Impersonate> false </Impersonate>
            <PrivateWorkspaceSupported> true </PrivateWorkspaceSupported>
        <RepositoryVersionManagerFactory>
            <JavaName>com.blazesoftware.repository.generic.version.NdNative
                RepositoryVersionManager </JavaName>
        </RepositoryVersionManagerFactory>

        </RepositoryVersionManagerConfig>
    </RepositoryConfig>

    <AllowUnusedValues> true </AllowUnusedValues>
</RomConfig>
```

For information about services you can include in a repository configuration file, see ["Writing a Repository Configuration File" on page 191](#).

A sample MongoDB repository configuration file `romConfig.cfg` is located in `<ADVISOR_HOME>/lib`. Save a backup copy of this file if you decide to edit its contents.

Creating or Removing a MongoDB or File Workspace

A workspace is a directory location on your local file system or a database table or database where you can obtain a copy of the latest files and resources for your project. Depending on whether or not you have a versioned repository and the type of versioning you are using, you can work in relative isolation from other users until you are ready to expose your work to other users who can have access to the latest files and resources by saving, updating or checking in your changes to repository.

Each user who needs access to a versioned MongoDB repository also needs their own workspace unless they are working in a MongoDB repository (versioning - shared) where they share a MongoDB workspace connected to a MongoDB repository with other users. For users to have access to a MongoDB repository (versioning -shared), you need to create a connection to the repository. Each user to have access to a MongoDB repository (versioning - private) or a MongoDB repository (BVS versioning - private local file), you need to create a workspace connected to the repository. The private workspaces for the MongoDB repository (versioning - private) are MongoDB type workspaces and are stored in the MongoDB database. The private workspaces for the MongoDB repository (BVS versioning - private local file) are File type workspaces that are stored on the local file system. The connection parameters you use to create the workspaces reflect the different workspace types and the location where they are stored.

You can create a workspace for MongoDB Repository (versioning - private) or a MongoDB Repository (BVS versioning - private local file) using the Blaze Advisor IDE or the `NdRomAdminUtil` utility:

- ["Creating a Workspace for a MongoDB Repository \(versioning - private\) Using the Blaze Advisor IDE" on page 365](#)
- ["Creating a Workspace for a MongoDB Repository \(BVS versioning - private local file\) Using the Blaze Advisor IDE" on page 368](#)
- ["Creating a Workspace for a MongoDB Repository \(versioning - private\) Using the NdRomAdminUtil Utility" on page 370](#)
- ["Creating a Workspace for a MongoDB Repository \(BVS versioning - private local file\) Using the NdRomAdminUtil Utility" on page 373](#)
- ["Removing a MongoDB or File Workspace Using the NdRomAdminUtil Utility" on page 376](#)

Creating a Workspace for a MongoDB Repository (versioning - private) Using the Blaze Advisor IDE

You can create a new private MongoDB type workspace for each user who needs to access a MongoDB Repository (versioning - private).

If you create workspaces for your users, if you want to provide the connection information for users to import, see “[Exporting a Blaze Advisor Workspace or Repository Connection](#)” on page 62.

To create a private MongoDB workspace

- 1 Select **Repository > New Workspace**.

Alternatively, from either the **File** menu or the Repository Explorer, select **New > Other** and expand the **Blaze Advisor** folder to choose **New Workspace**.

- 2 Select **MongoDB Repository (BVS private workspace)** from the Repository Type dropdown.
- 3 Enter a connection name for your workspace.
This name is used to identify the workspace connection in the IDE.
- 4 Click **Next**.
- 5 On the MongoDB Repository Settings section, enter the connection settings for the repository:
 - a Enter the **MongoDB URL** for the machine where the MongoDB database is running and enter a port number using the format <machineName or IP address>:<portNumber>. If you use the default MongoDB port number, entering it in the URL field is optional.
For example, 127.0.0.1:27017.
See “[MongoDB URL Connection Parameter](#)” on page 315.
 - b Select a **MongoDB authentication type** based on the authentication mode of the MongoDB database you want to use.
 - **None**
 - **MongoDB**
See “[MongoDB Authentication Type Connection Parameter](#)” on page 315.
 - c Enter a user name in the **User** field.
You need to ensure that the user name you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the user name you enter here is added to the system.user collection of your repository database. You may be prompted for administrative credentials if necessary to accomplish this.
 - d Enter a password name in the **Password** field.
You need to ensure that the user name you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the password you enter here is added to the system.user collection of your

repository database. You may be prompted for administrative credentials if necessary to accomplish this.

- e Enter a name in the **Repository name** field.

The name you specify here corresponds to the name used for the repository database and therefore cannot contain the following characters “/”, “\”, “.”, “*”, “<”, “>”, “:”, “|”, “?”, or double quotes.

- f (Optional) You can unselect the **Save database user and password** check box.

- This option is selected by default so that users can quickly and easily connect to the workspace, repository, or both by selecting a connection in the Manage Connections, Import, or Export wizards. The password is encrypted by default. See [“Saving User Name and Passwords” on page 43](#).
- If this option is unselected, user names and passwords are not persisted in the system, however it also means that users will be prompted whenever a connection to the workspace, repository, or both needs to be made.

- 6 In the Workspace Settings section enter the workspace connection information:

- a Enter a name in the **Workspace Name** field.

The name you specify here corresponds to the name used for the workspace database and therefore cannot contain the following characters “/”, “\”, “.”, “*”, “<”, “>”, “:”, “|”, “?”, or double quotes.

- b Enter a user name in the **User** field.

You need to ensure that the user name you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the user name you enter here is added to the system.user collection of your workspace database. You may be prompted for administrative credentials if necessary to accomplish this.

- c Enter a password name in the **Password** field.

You need to ensure that the password you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the password you enter here is added to the system.user collection of your workspace database. You may be prompted for administrative credentials if necessary to accomplish this.

- d (Optional) You can select the **Save workspace user and password** check box to use a link to quickly and easily connect to the workspace or repository in Blaze Advisor. See [“Saving User Name and Passwords” on page 43](#).

7 Click **Finish**.

You connect to the workspace by opening it in Blaze Advisor. See “[Viewing the Repository or Workspace in the Blaze Advisor IDE](#)” on page 381.

After you connect to the workspace, you may need to explicitly update your workspace to see the latest versions of all files.

In future, if you need to create a new connection, see “[Connecting to a MongoDB Repository Using the Blaze Advisor IDE](#)” on page 378.

Creating a Workspace for a MongoDB Repository (BVS versioning - private local file) Using the Blaze Advisor IDE

You can create a new private local file type workspace for each user who needs to access a MongoDB Repository (BVS versioning - private local file).

If you create workspaces for your users, if you want to provide the connection information for users to import, see “[Exporting a Blaze Advisor Workspace or Repository Connection](#)” on page 62.

To create a private local file workspace

1 Select **Repository > New Workspace**.

Alternatively, from either the **File** menu or the Repository Explorer, select **New > Other** and expand the **Blaze Advisor** folder to choose **New Workspace**.

2 Select **MongoDB Repository (BVS private local file workspace)** from the Repository Type dropdown.

3 Enter a connection name for your workspace.

This name is used to identify the workspace connection in the IDE.

4 Click **Next**.

5 On the MongoDB Repository Settings section, enter the connection settings for the repository:

- a Enter the **MongoDB URL** for the machine where the MongoDB database is running and enter a port number using the format `<machineName or IP address>:<portNumber>`. If you use the default MongoDB port number, entering it in the URL field is optional.

For example, `127.0.0.1:27017`.

See “[MongoDB URL Connection Parameter](#)” on page 315.

- b Select a **MongoDB authentication type** based on the authentication mode of the MongoDB database you want to use.

■ **None**

■ **MongoDB**

See “[MongoDB Authentication Type Connection Parameter](#)” on page 315.

c Enter a user name in the **User** field.

You need to ensure that the user name you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the user name you enter here is added to the system.user collection of your repository database. You may be prompted for administrative credentials if necessary to accomplish this.

d Enter a password name in the **Password** field.

You need to ensure that the password you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the password you enter here is added to the system.user collection of your repository database.

e Enter a name in the **Repository name** field.

The name you specify here corresponds to the name used for the repository database and therefore cannot contain the following characters "/", "\", "", "*", "<", ">", ".", "|", "?" or double quotes.

f (Optional) You can unselect the **Save database user and password** check box.

- This option is selected by default so that users can quickly and easily connect to the workspace, repository, or both by selecting a connection in the Manage Connections, Import, or Export wizards. The password is encrypted by default. See ["Saving User Name and Passwords" on page 43](#).
- If this option is unselected, user names and passwords are not persisted in the system, however it also means that users will be prompted whenever a connection to the workspace, repository, or both needs to be made.

6 In the Workspace Settings section enter the workspace connection information:

a Enter or browse to a local folder in the **Local Folder** field.

b Enter a user name in the **User** field.

You need to ensure that the user name you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode).

c Enter a password name in the **Password** field.

You need to ensure that the password you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode).

d (Optional) You can select the **Save workspace user and password** check box to use a link to quickly and easily connect to the workspace or repository in Blaze Advisor. See ["Saving User Name and Passwords" on page 43](#).

7 Click **Finish**.

You connect to the workspace by opening it in Blaze Advisor. See “[Viewing the Repository or Workspace in the Blaze Advisor IDE](#)” on page 381.

After you connect to the workspace, you may need to explicitly update your workspace to see the latest versions of all files.

In future, if you need to create a new connection, see “[Connecting to a MongoDB Repository Using the Blaze Advisor IDE](#)” on page 378.

Creating a Workspace for a MongoDB Repository (versioning - private) Using the NdRomAdminUtil Utility

You can create a new private MongoDB type workspace for each user who needs to access a MongoDB Repository (versioning - private) using the `NdRomAdminUtil` utility by preparing the following files:

- If you created the MongoDB repository (versioning - private) using the `NdRomAdminUtil` utility, you may already have a repository connection configuration file for the MongoDB repository connection you want to use to create your workspaces, for more information about the contents of a repository connection configuration file, see “[Repository Configuration File for a MongoDB Repository \(versioning - private\)](#)” on page 353.
- “[Creating a Workspace Connection Configuration File for the MongoDB Repository \(versioning - private\)](#)” on page 370
- “[Creating a Batch File or Running the NdRomAdminUtil Utility to Create a Workspace for a MongoDB Repository \(versioning - private\)](#)” on page 373

After you create workspaces for your users, you need to create a connection in the Blaze Advisor IDE, see “[Connecting to a MongoDB Repository Using the Blaze Advisor IDE](#)” on page 378.

Creating a Workspace Connection Configuration File for the MongoDB Repository (versioning - private)

You write a workspace connection configuration file for a MongoDB workspace with information including the connection class, the MongoDB URL, the repository connection name, and the repository name. This is similar to the information you would enter in the New Workspace wizard in the Blaze Advisor IDE.

You can create the workspace connection configuration for your MongoDB Repository (versioning - private) using a text editor and save the file with a `.cfg` extension. The connection configuration file contains the following tag structure and values:



Note For reasons of brevity, a pair of element tags will be represented by the following notation, `<example>`.

Tag	Description
<?xml version="1.0" encoding="UTF-8" ?>	Processing information that contains the XML version and character-encoding used.
<RepositoryConnection/>	The RepositoryConnection tag is used when you need to create a repository connection. All other connection tags in this table are nested within this tag.
<Factory/>	The Factory tag is used when you want to create an instance of a class. To create an instance of a connection for a MongoDB non-versioned repository, you use the fully qualified class name: com.blazesoft.repository.mongodb.NdMongoRepositoryConnection.
<MongoDBAuthenticationType/>	The authentication type of the MongoDB instance. Available values are: <ul style="list-style-type: none"> ■ 0 -- None ■ 1 -- MongoDB See " MongoDB Authentication Type Connection Parameter " on page 315.
<MongoDBUrl/>	The URL of the MongoDB instance. See " MongoDB URL Connection Parameter " on page 315. Example: 127.0.0.1:27017
<Name/>	The name is used to specify the connection name listed in the Manage Connections wizard in the Blaze Advisor IDE.
<RepositoryName/>	The name you specify here corresponds to the name used for the database and therefore cannot contain the following characters "/", "\", ".", "*", "<", ">", ":", " ", "?", or double quotes.
<User/>	<p>The user name.</p> <ul style="list-style-type: none"> ■ If the MongoDB database you are connecting to is running in authenticated mode and you have connected using the MongoDB authentication type, the user name is added to the system.user collection and is used by MongoDB to authenticate the user. The authorization manager also uses the same user name to authenticate the user for access to items in the repository or workspace. ■ If the MongoDB database you are connecting to is running in unauthenticated mode and you have connected using the None authentication type, the MongoDB does not authenticate the user. The authorization manager uses the user name to authenticate the user for access to items in the repository or workspace.

Tag	Description
<Password/>	<p>The password.</p> <ul style="list-style-type: none"> ■ If the MongoDB database you are connecting to is running in authenticated mode and you have connected using the MongoDB authentication type, the password is added to the system.user collection and is used by MongoDB to authenticate the user. The authorization manager also uses the same password to authenticate the user for access to items in the repository or workspace. ■ If the MongoDB database you are connecting to is running in unauthenticated mode and you have connected using the None authentication type, the MongoDB does not authenticate the user. The authorization manager uses the password to authenticate the user for access to items in the repository or workspace.
<SharedWorkspace/>	Value is false for this MongoDB repository configuration.
<UseSSLEnabled/>	Value is false. This value is true only if you are using a Secure Sockets Layer (SSL) available only with MongoDB Enterprise Edition. See “ SSL Connections and MongoDB Repositories ” on page 383.
<WorkspaceConnection/>	The value is true. This is a workspace connection.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<RepositoryConnection>
    <Factory>com.blazesoft.repository.mongodb.NdMongo
        RepositoryConnection </Factory>
    <MongoDBAuthenticationType>1</MongoDBAuthenticationType>
    <MongoDBUrl> 127.0.0.1:27017 </MongoDBUrl>
    <Name>MongoWsConnection</Name>
    <RepositoryName>MongoWs01</RepositoryName>
    <User>user1</User>
    <Password>user1</Password>
    <SharedWorkspace>false</SharedWorkspace>
    <UseSSLEnabled>false</UseSSLEnabled>
    <WorkspaceConnection>true</WorkspaceConnection>
</RepositoryConnection>
```

A sample MongoDB workspace connection configuration file `mongoWorkspace.cfg` is located in `<ADVISOR_HOME>/lib`. You may need to edit this file to create a workspace connection configuration. Save a backup copy of this file if you decide to edit its contents.

Creating a Batch File or Running the NdRomAdminUtil Utility to Create a Workspace for a MongoDB Repository (versioning - private)

You can either run the `NdRomAdminUtil` utility directly from the command line or you can write a batch file referencing the utility to create a repository.

Command, Argument, or Option	Description
<code>createWorkspace</code>	Command to create a workspace.
<code>-repositoryConnection</code>	Argument takes the path to the location of the repository connection configuration file (.cfg). This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
<code>-workspaceConnection</code>	Argument takes the path to the location of the Workspace connection configuration file (.cfg). This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
<code>-verbose</code>	Option prints out all details messages into the command console.

Example of a batch file to create a workspace using pathnames:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil createWorkspace
-repositoryConnection "c:/Repo_Connections/ConnectionFiles/
DefaultVerRepo.cfg"
-workspaceConnection "c:/Repo_Connections/ConnectionFiles/
DefaultVerWksp.cfg"
-verbose
```

Creating a Workspace for a MongoDB Repository (BVS versioning - private local file) Using the NdRomAdminUtil Utility

You can create a new private MongoDB type workspace for each user who needs to access a MongoDB Repository (versioning - private) using the `NdRomAdminUtil` utility by preparing the following files:

- If you created the MongoDB repository (versioning - private) using the `NdRomAdminUtil` utility, you may already have a repository connection configuration file for the MongoDB repository connection you want to use to create your workspaces, for more information about the contents of a repository connection configuration file, see "[Repository Configuration File for a MongoDB Repository \(BVS versioning - private local file\)](#)" on page 363.
- "[Creating a Workspace Connection Configuration File for a MongoDB Repository \(BVS versioning - private local file\)](#)" on page 374
- "[Creating a Batch File or Running the NdRomAdminUtil Utility to Create a Workspace for a MongoDB Repository \(BVS versioning - private local file\)](#)" on page 376

Creating a Workspace Connection Configuration File for a MongoDB Repository (BVS versioning - private local file)

You write a workspace connection configuration file for a file workspace with information including the connection class, the name of the repository and the workspace folder location. This is similar to the information you would enter in the New Workspace wizard in the Blaze Advisor IDE.

You can create the workspace connection configuration for your MongoDB Repository (BVS versioning - private local file) using a text editor and save the file with a `.cfg` extension. The connection configuration file contains the following tag structure and values:



Note For reasons of brevity, a pair of element tags will be represented by the following notation, `<example/>`.

Tag	Description
<code><?xml version="1.0" encoding="UTF-8" ?></code>	Processing information that contains the XML version and character-encoding used.
<code><RepositoryConnection/></code>	The RepositoryConnection tag is used when you need to create a repository connection. All other connection tags in this table are nested within this tag.
<code><Factory/></code>	The Factory tag is used when you want to create an instance of a class. To create a file workspace repository, you use the fully qualified class name: <code>com.blazesoft.repository.file.NdFileRepository Connection</code> .
<code><RepositoryName/></code>	The name you specify here corresponds to the name used for the repository database you want to use for your connection and therefore cannot contain the following characters <code>/</code> , <code>\</code> , <code>,</code> , <code>*</code> , <code><</code> , <code>></code> , <code>.</code> , <code>,</code> , <code> </code> , <code>?</code> , or double quotes.

Tag	Description
<User/>	<p>The user name.</p> <ul style="list-style-type: none"> ■ If the MongoDB database you are connecting to is running in authenticated mode and you have connected using the MongoDB authentication type, the user name is added to the system.user collection and is used by MongoDB to authenticate the user. The authorization manager also uses the same user name to authenticate the user for access to items in the repository or workspace. ■ If the MongoDB database you are connecting to is running in unauthenticated mode and you have connected using the None authentication type, the MongoDB does not authenticate the user. The authorization manager uses the user name to authenticate the user for access to items in the repository or workspace.
<Password/>	<p>The password.</p> <ul style="list-style-type: none"> ■ If the MongoDB database you are connecting to is running in authenticated mode and you have connected using the MongoDB authentication type, the password is added to the system.user collection and is used by MongoDB to authenticate the user. The authorization manager also uses the same password to authenticate the user for access to items in the repository or workspace. ■ If the MongoDB database you are connecting to is running in unauthenticated mode and you have connected using the None authentication type, the MongoDB does not authenticate the user. The authorization manager uses the password to authenticate the user for access to items in the repository or workspace.



Note To avoid errors, do not leave any trailing spaces on any lines in your file. If you choose to copy and paste this example, please be aware that it was formatted to fit the page.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>

<RepositoryConnection>
  <Factory
    com.blazesoft.repository.file.NdFileRepositoryConnection
  </Factory>
  <RepositoryFolder> C:\Wksps\MongoDBFileWkspDir</RepositoryFolder>
  <RepositoryName>MongoDB5a1 </RepositoryName>
  <User>user1</User>
  <Password>user1</Password>
</RepositoryConnection>
```

A sample file workspace connection configuration file `FileWorkspace.cfg` is located in `<ADVISOR_HOME>/lib`. You may need to edit this file to create a workspace connection configuration. Save a backup copy of this file if you decide to edit its contents.

Creating a Batch File or Running the NdRomAdminUtil Utility to Create a Workspace for a MongoDB Repository (BVS versioning - private local file)

You can either run the `NdRomAdminUtil` utility directly from the command line or you can write a batch file referencing the utility to create a repository.

Command, Argument, or Option	Description
<code>createWorkspace</code>	Command to create a workspace.
<code>-repositoryConnection</code>	Argument takes the path to the location of the repository connection configuration file (.cfg). This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
<code>-workspaceConnection</code>	Argument takes the path to the location of the Workspace connection configuration file (.cfg). This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
<code>-verbose</code>	Option prints out all details messages into the command console.

Example of a batch file to create a workspace using pathnames:

```
java com.blazesoft.template.repository.admin.NdRomAdminUtil createWorkspace
-repositoryConnection "c:/Repo_Connections/ConnectionFiles/
DefaultVerRepo.cfg"
-workspaceConnection "c:/Repo_Connections/ConnectionFiles/
DefaultVerWksp.cfg"
-verbose
```

Removing a MongoDB or File Workspace Using the NdRomAdminUtil Utility

You can remove a workspace using the `NdRomAdminUtil` utility. You can use the connection configuration file for the MongoDB repository and the configuration file for the MongoDB workspace or File workspace connection.

If you used the `NdRomAdminUtil` utility to create the MongoDB repository (versioning - private) or the MongoDB repository (BVS versioning - private local file), you can reuse the repository connection configuration file you created to use with the utility. If you did not use the `NdRomAdminUtil` utility to create the MongoDB repository, you need to create a repository connection configuration file. See:

- “[Creating a MongoDB Repository \(BVS versioning - private\) Using the NdRomAdminUtil utility](#)” on page 350
Or
- “[Creating a MongoDB Repository \(BVS versioning - private local file workspace\) Using the NdRomAdminUtil utility](#)” on page 360

If you created the private MongoDB workspaces or the private local file workspaces using the `NdRomAdminUtil` utility, you can reuse the workspace connection configuration file. If you did not use the `NdRomAdminUtil` utility to create the workspace, you need to create a workspace connection configuration file. See:

- “[Creating a Workspace for a MongoDB Repository \(versioning - private\) Using the NdRomAdminUtil Utility](#)” on page 370
Or
- “[Creating a Workspace for a MongoDB Repository \(BVS versioning - private local file\) Using the NdRomAdminUtil Utility](#)” on page 373

After you have located or prepared the repository connection configuration file and the workspace connection configuration file, you can use the `NdRomAdminUtil` utility `removeWorkspace` command to remove the workspace, see “[Creating a Batch File or Running the NdRomAdminUtil Utility to Remove a MongoDB Workspace or a File Workspace Connected to a MongoDB Repository](#)” on page 377.

Creating a Batch File or Running the NdRomAdminUtil Utility to Remove a MongoDB Workspace or a File Workspace Connected to a MongoDB Repository

To remove a workspace using the `NdRomAdminUtil` utility, you can use a batch file or command prompt. In the file or prompt, you add the following command, argument, and argument values.

Command, Argument, or Option	Description
<code>removeWorkspace</code>	Command to remove a workspace.
<code>-repositoryConnection</code>	Argument takes the path to the location of the repository connection configuration file (.cfg). This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
<code>-workSpaceConnection</code>	Argument takes the path to the location of the Workspace connection configuration file (.cfg). This file contains information such as the name of the repository, the location of the repository directory, and user name and password if applicable.
<code>-verbose</code>	Option prints out all details messages into the command console.

Example of a batch file to remove a File workspace using pathnames:

```
java com.blazesoftware.template.repository.admin.NdRomAdminUtil removeWorkspace
-repositoryConnection "C:/Repo_Connections/ConnectionFiles/
RepoConnectionConfig.cfg"
-workspaceConnection "c:/Repo_Connections/ConnectionFiles/
WorkspaceConnectionConfig.cfg"
-verbose
```

Connecting to a MongoDB Repository Using the Blaze Advisor IDE

A repository or workspace connection allows you to access the items in a workspace or repository in the Blaze Advisor IDE. When you create a workspace or repository using the Blaze Advisor IDE, the creation process automatically creates a connection to the workspace or repository in the IDE.

When you create the workspace or repository using the `NdRomAdminUtil` utility, you need to create a connection to the workspace or repository in the Blaze Advisor IDE before you can access the workspace or repository in the IDE. Likewise, if you need to connect to an existing non-versioned MongoDB repository, a versioned MongoDB repository with a shared workspace, or a versioned MongoDB repository with private MongoDB workspaces or private local file workspaces, you need to create a new connection in the IDE for each user to access the repository or shared workspace or their private workspace. See ["Creating a New Connection for a MongoDB or File Workspace or Repository Using the Blaze Advisor IDE" on page 378](#).

If you are using a MongoDB repository (versioning - private) or a MongoDB repository (BVS versioning - private local file) you need to create a new workspace for each user before you can create a connection to the workspace in the Blaze Advisor IDE. See ["Creating or Removing a MongoDB or File Workspace" on page 365](#).

After you have created the workspace or repository connection in the IDE, you can export the connection information in a file and send it to your user(s), see ["Exporting a Blaze Advisor Workspace or Repository Connection" on page 62](#).

Your users can import the connection file into their own Blaze Advisor IDE installation and connect to the MongoDB repository, see ["Importing a Blaze Advisor Workspace or Repository Connection" on page 61](#).

Creating a New Connection for a MongoDB or File Workspace or Repository Using the Blaze Advisor IDE

You can use the New Repository Connection dialog to create a connection to an existing MongoDB workspace or repository that you created using the Blaze Advisor IDE or the `NdRomAdminUtil` utility.

To create a new connection for a MongoDB or File workspace or repository

- 1 Select **Repository > New Repository Connection** or **Repository > Manage Connections** and click **New**.

Alternatively, from either the **File** menu or the Repository Explorer, select **New > Other** and expand the **Blaze Advisor** folder to choose **New Repository Connection**.

- 2 Enter a **Connection Name**.

This name will display in the Manage Connections wizard.

- 3 Select a workspace type.

The type of a workspace or repository can be File, Database, or MongoDB.

For MongoDB repository configurations, you can choose between File or MongoDB.

- If you are connecting to a MongoDB repository that is one of the following:

- MongoDB Repository (no versioning)
- MongoDB Repository (with Authorization manager)
- MongoDB Repository (versioning - private)
- MongoDB Repository (versioning - shared)

Select **MongoDB** as the workspace type.

- If you are connecting to a MongoDB repository that is MongoDB Repository (BVS versioning - private local file), select **File** as the workspace type.

- 4 Click **Next**.

- 5 If you are connecting to a MongoDB workspace type, the connection parameters you enter are based on the MongoDB repository configuration. If you are connecting to a File workspace type, skip to Step 6.

- a Enter the **MongoDB URL** for the machine where the MongoDB database is running and enter a port number using the format <machineName or IP address>:<portNumber>. If you use the default MongoDB port number, entering it in the URL field is optional.

For example, 127.0.0.1:27017.

See "[MongoDB URL Connection Parameter](#)" on page 315.

- b Select a **MongoDB authentication type** based on the authentication mode of the MongoDB database connection.

- **None**

- **MongoDB**

See "[MongoDB Authentication Type Connection Parameter](#)" on page 315.

- c Enter your user name in the **User** field.

You need to ensure that the user name you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the user name you enter here must be present in the system.user collection of your repository database.
 - d Enter your password name in the **Password** field.

You need to ensure that the password you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode). If your MongoDB is running with authenticated mode, the password you enter here must be present in the system.user collection of your repository database.
 - e Enter the Workspace name in the **Repository name** field.

The name you specify here corresponds to the name used for the repository database and therefore cannot contain the following characters "/", "\", ".", "*", "<", ">", ":" , "|", "?" , or double quotes.
 - f (Optional) You can unselect the **Save database user and password** check box.
 - This option is selected by default so that users can quickly and easily connect to the workspace, repository, or both by selecting a connection in the Manage Connections, Import, or Export wizards. The password is encrypted by default. See "[Saving User Name and Passwords](#)" on page 43.
 - If this option is unselected, user names and passwords are not persisted in the system, however it also means that users will be prompted whenever a connection to the workspace, repository, or both needs to be made.
- 6 If you are connecting to a file workspace type connected to a MongoDB Repository (BVS versioning - private local file).
- a Enter or browse to the location of the local file workspace.

Because the workspace directory contains the connection to the MongoDB repository, as soon as you select the workspace location, the page is refreshed with the following fields, check box and button:
 - b Enter the user name for the workspace in the **User** field.

You need to ensure that the user name you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode).
 - c Enter the password for the workspace in the **Password** field.

You need to ensure that the password you enter here is the same one used with your versioning service and the MongoDB database (if running with authenticated mode).

- d (Optional) You can select the **Save user and password** check box. See "[Saving User Name and Passwords](#)" on page 43.
 - e (Optional) Click **Test Connection** to verify if entered your user name and password correctly
- 7 Click **Finish**.
- You see a message confirming that the connection has been created.
- 8 Click **OK**.

Viewing the Repository or Workspace in the Blaze Advisor IDE

When you create a repository or workspace or create a connection to a repository or workspace in the Blaze Advisor IDE, the connection to this new repository is available to you in the Blaze Advisor IDE. You can connect to the repository or workspace and view its contents in the Repository Explorer view.

- 1 Select **Window > Show View > Repository Explorer** and if the repository you just created is not displayed, select **Select a different repository connection** from the explorer toolbar.
- 2 Select the Repository Connection name from the list, this is the name you entered for the Repository Nickname when you created the repository and click **OK**.

Converting to a Cloud Compatible MongoDB Repository

If you are already using a Blaze Advisor repository, there are several ways you can move the contents of your existing Blaze Advisor repository into a MongoDB repository. The Import and Export commands in the Blaze Advisor IDE or the **NdRomAdminUtil** utility allow you to move the source workspace contents to a target MongoDB workspace where you can check in the files. For example, if you have a File repository with no versioning and you want to use a MongoDB Repository (BVS versioning - private local file), you can export the source workspace contents to the target file workspace and check in the files into the MongoDB repository. From that point, you can connect to the file workspace to continue working on the files. At some point later when you want to use a cloud compatible repository, you can convert the MongoDB Repository (BVS versioning - private local file) to a MongoDB Repository (versioning - private) that stores its private workspaces in the MongoDB. See "["Converting a MongoDB Repository \(BVS versioning - private local file\) to a MongoDB Repository \(BVS versioning - private\)"](#) on page 382."

Converting to a MongoDB repository (BVS versioning - private local file) from another repository type can be an intermediate step. For example, if you have a Database repository and you want to use a MongoDB Repository (BVS versioning - private local file), you can use the same commands in the Blaze Advisor IDE or the **NdRomAdminUtil** utility to convert the contents created in the Database repository to use a file workspace connected to a MongoDB repository before converting it to a MongoDB repository where the workspaces are also stored in MongoDB.

You could also convert directly to a MongoDB BVS private repository with a private workspace hosted in MongoDB by creating the MongoDB repository and workspace and then importing directly into the MongoDB workspace from the source workspace.

Connections to both the source and target workspace must exist for you to convert the workspace contents successfully. In the Blaze Advisor IDE, to export a workspace, you can use either the File > Export and expand the Blaze Advisor Repository Administration folder and expand the Advisor Workspace folders to select, From another Blaze Advisor workspace or to import a workspace, you can use File > Import and expand the Blaze Advisor and Workspace folders to select, Advisor Workspace to another Advisor Workspace, as long as you have access to both the source and target workspace or repository connections. You can use the `NdRomAdminUtil` utility, `exportImportWorkspace` command after you create the workspace connection configuration file for both the source and target workspaces.

If the target workspace is a MongoDB private local file workspace, you move the contents of the source workspace to the directory location of the file workspace. In both cases because you are using the workspaces to convert the container type you are working in, you need to check in the files after export. For example, if you have a File CVS repository and you want to convert to use a MongoDB repository with a private local file workspace, assuming that both the source and target workspaces are file type, you can move the contents from your source workspace to the target workspace and check in the changes. Please note that the version history is not persisted to the target repository connection.

You can use one of these methods to move your workspace contents:

- “Importing a Blaze Advisor Contents from Another Blaze Advisor Workspace” in *DevelopingRuleProjects.pdf*
- “Exporting a Blaze Advisor Contents to Another Blaze Advisor Workspace” in *DevelopingRuleProjects.pdf*
- “[Exporting and Importing a Workspace Using the Utility](#)” on page 75

Converting a MongoDB Repository (BVS versioning - private local file) to a MongoDB Repository (BVS versioning - private)

The MongoDB Repository (BVS versioning - private local file) is not cloud compatible because it uses local file workspaces. If you are using a MongoDB Repository (BVS versioning - private local file) and cloud compatibility becomes a requirement, you can convert your repository to a MongoDB Repository (versioning - private) where BVS versioning and the private workspaces are stored in the MongoDB.

You can convert your existing repository to a cloud compatible repository by creating a new MongoDB workspace connecting to your existing MongoDB repository using the Blaze Advisor IDE or the `NdRomAdminUtil` utility. See “[Creating or Removing a MongoDB or File Workspace](#)” on page 365.



Important Prior to creating the new workspace, ensure that any changes you have made to items in your file workspace have been checked in.

If the MongoDB repository (BVS versioning - private local file) had version history prior to conversion to a MongoDB repository (versioning - private), the version history is persisted. If you converted from another repository type to a MongoDB repository (BVS versioning - private local file) as an intermediate step, your version history is not preserved. If you generated an RMA prior to conversion, see ["Regenerating an RMA After Converting to a MongoDB Repository" on page 383](#).

Regenerating an RMA After Converting to a MongoDB Repository

If you had generated RMA workspaces prior to converting to a MongoDB Repository (BVS versioning - private), after the conversion you will need to regenerate your RMA in order to use cloud compatible RMA workspaces.

SSL Connections and MongoDB Repositories

Secure Socket Layer (SSL) is a protocol for encrypting information over the Internet. Blaze Advisor MongoDB repositories using SSL require that you have installed MongoDB Enterprise Edition. For more information about the MongoDB Enterprise Edition, see <http://www.mongodb.com/subscription/downloads>.

This section includes the following topics:

- ["Configuring MongoDB to use SSL" on page 383](#)
- ["Testing MongoDB with a Self-signed SSL Certificate" on page 384](#)

Configuring MongoDB to use SSL

The following assumes that you have already installed MongoDB Enterprise Edition. Before you can use SSL, you need to create a .pem file that contains a SSL certificate file signed by a Certificate Authority (CA) and a separate .pem file corresponding CA private key file.



Note SSL encryption only mode is supported in this release. There is currently no support for MongoDB servers that are setup to require client certificate validation.

To enable MongoDB to use SSL

- 1 Select the **Start menu > All Programs > Accessories** and then right-click the Command Prompt to select **Run as Administrator**.
- 2 If the SSL certificate and private key produced by the external CA authority were provided in a .pfx file, the .pfx file needs to be converted into a .pem file for use in the MongoDB using openssl.
`openssl pkcs 12 -in <sslKeyFileName>.pfx -out <sslKeyFileName>.pem`
If the private key in the .pfx file is password protected, you will get prompted for a password three times. If you do not know the password, you need to contact your CA authority.

- 3 In order for the above .pem file containing the SSL certificate and key to work with MongoDB, there must be another .pem file that must contain the full certificate chain back to the issuing CA authority, the certificates for this chain need to be added to another .pem file in the correct order. For example:

```
cat "AddTrust External CA Root.cer" "Name of Secure CA Server.cer" >
CAchainFileName.pem
```

- 4 Add the following lines to the mongod.cfg file to configure MongoDB to use the .pem files that contains the SSL certificate and the CA chain:

```
sslOnNormalPorts=true
sslPEMKeyFile=<location of sslKeyFileName.pem file>
sslPEMKeyPassword=<passwordForSSLKeyFile>
sslCAFile=<location of CAchainFileName.pem file>
```

For example if the .pem files are placed in the Mongo installation directory:

```
sslOnNormalPorts=true
sslPEMKeyFile=mongoInstallDir/sslKeyFileName.pem
sslPEMKeyPassword=passwordForSSLKeyFile
sslCAFile=mongoInstallDir/CAchainFileName.pem
```



Note The above examples use the original MongoDB configuration file format. If you are using the new YAML-based configuration file format introduced in MongoDB 2.6, then these entries would have to have the format below:

```
net:
  ssl:
    sslOnNormalPorts=true
    PEMKeyFile=mongoInstallDir/sslKeyFileName.pem
    PEMKeyPassword=passwordForSSLKeyFile
    CAFile=mongoInstallDir/CAchainFileName.pem
```

- 5 You can connect to the MongoDB database by selecting **Use SSL (MongoDB Enterprise Edition Required)**.

Testing MongoDB with a Self-signed SSL Certificate

The following assumes that you have already installed MongoDB Enterprise Edition.

Before you can use SSL, you need to create a .pem file that contains a SSL certificate file signed by a certificate authority and a corresponding private key file. If you want to test MongoDB with SSL enabled prior to obtaining a SSL certificate signed by a certificate authority you can create a self-signed certificate and private key file.



Note SSL encryption only mode is supported in this release. There is currently no support for MongoDB servers that are setup to require client certificate validation.

To test MongoDB with a Self-signed SSL certificate

- 1 Select the **Start menu > All Programs > Accessories** and then right-click the Command Prompt to select **Run as Administrator**.
- 2 Enter the following command to create a new, self-signed certificate with no passphrase that is valid for 365 days.
`cd /etc/ssl/openssl req -new -x509 -days 365 -nodes -out mongodb-cert.crt -keyout mongodb-cert.key`

- 3** After you have generated the certificate, concatenate the certificate and private key to a .pem file, for example:

```
cat mongodb-cert.key mongodb-cert.crt > mongodb.pem
```

Add the following lines to the mongod.cfg file to enable SSL.

```
sslOnNormalPorts=true
sslPEMKeyFile=<location of .pem file>
```

The certificate created by the previous steps is a self-signed certificate rather than one that is properly signed from a known certificate authority. Therefore, you need to do some additional configuration on the Java client machine to register this certificate as a trusted certificate.

- 4** Copy the mongodb-cert.crt certificate file to the client machine.

- 5** Execute the following command in one single line:

```
$JAVA_HOME/bin/keytool -import -v -trustcacerts -alias
MongoDBTestCertificate -file mongodb-cert.crt -keystore
$JAVA_HOME/jre/lib/security/jssecacerts -storepass changeit
```

- 6** You can connect to the MongoDB database by checking **Use SSL (MongoDB Enterprise Edition Required)**.

RMA Generation for MongoDB Repositories and Workspaces

When you generate an RMA for a project stored in a repository with versioning, you specify a location for the RMA workspaces that are created for each user logging into the RMA. These file workspaces are connected to the repository and are created specifically for use with the RMA and are separate from the workspaces users may have been using when working in the Blaze Advisor IDE.

When you generate an RMA for a project stored in a MongoDB BVS repository, the RMA Generator creates new workspaces of the same type that was used by the connection. If you are using a Blaze Advisor MongoDB repository and you are using private workspaces, you can decide what type of RMA workspaces you want to use before you generate the RMA.

- If you generate an RMA from a project in a MongoDB BVS repository with private local file workspaces, the RMA Generator creates file workspaces for use with the RMA. You can specify the location for the file workspaces in the Workspaces directory in the RMA Generator wizard.
- If you generate an RMA from a project in a MongoDB BVS repository with private MongoDB workspaces stored in the MongoDB repository, the RMA Generator wizard creates an RMA workspace in MongoDB for use with the RMA. You can control the names of the RMA MongoDB workspaces, see ["Workspace Base Name Setting for MongoDB Workspaces in the RMA Generator" on page 386](#).

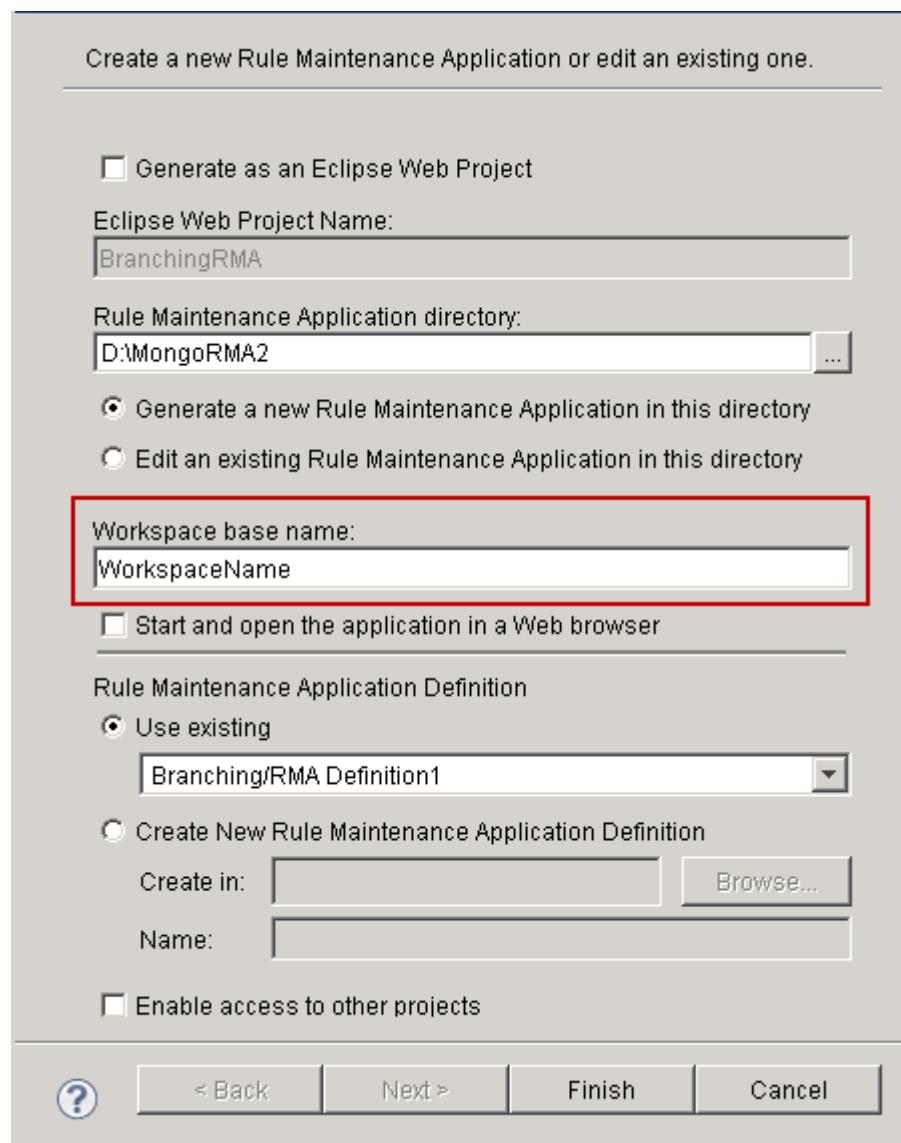
If you generate an RMA from a project stored in a non-versioned MongoDB repository, a MongoDB repository (with Authorization Manager), or a MongoDB repository (BVS -

shared) you and your users would log on to connect to the same workspace in MongoDB that you used in the Blaze Advisor IDE.

If you want the RMA to be cloud compatible, you need the RMA to create workspaces hosted in MongoDB rather than a local file-based workspaces. If you have been developing your project using a private local file workspace, you need to create a new MongoDB private workspace and connect it to your existing MongoDB repository to convert it to a cloud compatible repository. See ["Converting to a Cloud Compatible MongoDB Repository" on page 381](#).

Workspace Base Name Setting for MongoDB Workspaces in the RMA Generator

When you generate the RMA from a project in MongoDB Repository (versioning - private) where the workspaces are hosted in the MongoDB database, the Workspaces directory field in the RMA Generator wizard is replaced by a Workspace base name field. The base name of the workspace that will be created by the RMA is displayed in the Workspace base name field. The Workspace base name takes the name of the MongoDB workspace used by the user who generated the RMA and uses it as the prefix for any RMA workspaces created when other users log onto the RMA.



You can either use the name of your workspace as the Workspace base name or enter another name. The name used for the Workspace base name is used as the prefix for all RMA workspaces created as soon as a user logs onto the generated RMA.

If you generated the RMA and you used the default settings on the Workspaces tab in the RMAD, when you log onto the RMA as the default user, a new workspace is not created for you in the MongoDB, instead you connect to the same MongoDB workspace you used when you connected to the IDE. If you log on as a new user, a new workspace is created automatically.

The format for the workspace names created by the RMA is:

`workspaceBaseNameUserName`

- The `workspaceBaseName` is the value entered in the Workspace base name field in the RMA Generator wizard.

- The **UserName** is the user name entered when logging onto the RMA

For example if you enter **RMAProjDev_** in the **Workspace Base Name** field, when one of your users logs onto the RMA as **user1**, a new RMA workspace is created in the MongoDB database called **RMAProjDev_user1**. Whenever user1 logs onto the RMA, they will use this workspace.

You can customize the workspace names using the values of the connection parameters by using **%connParamName%** to specify a pattern for how the **workspaceBaseName** value is used to name a MongoDB RMA workspace at runtime.

Example:

MyWorkspaceFor_%com.blazesoftware.repository.username%Workspace

Or

RMAWksp_%username%_Workspace

For example, if you used **RMAWkspFor_%username%** in your Workspace Base Name field in the RMA Generator, when userA logs into the RMA, a MongoDB RMA workspace called **RMAWkspFor_userA** workspace is added to the MongoDB database. If another user logs on as userB, a MongoDB RMA workspace called **RMAWkspFor_userB** workspace is added.

Implementing an Authorization Manager

FICO® Blaze Advisor® decision rules management system provides an authorization manager interface that you can use to write a custom class to enforce authorization security controls for a repository and project contents. If you are using an LDAP or database server for your Blaze Advisor repository implementation, you may already be able to provide adequate security controls for your repository; however, if you are unable to achieve the necessary security controls, you can write a custom authorization manager class. When a repository is configured with an authorization service, also known as repository-level authorization, you can perform basic authentication based on user logon information. The authenticated logon information obtained is used to determine what operations can be performed on repository entries by users.

If you want to implement access control for an RMA without implementing repository-level authorization as described here, you can add Life Cycle Management components to your RMA, see “Enabling Life Cycle Management” in *DevelopingRuleMaintenanceApplications.pdf* and “Managing Instances Using Life Cycle Management” in *DevelopingRuleMaintenanceApplications.pdf*.

If you want to implement IP protection to your repository contents, see “[Implementing Content Protection for Repository Items](#)” on page 413.

This section contains the following topics:

- [“Writing a Custom Authorization Manager” on page 390](#)
- [“Creating a Repository With a Custom Authorization Manager Class” on page 402](#)
- [“Connecting to a Repository With an Authentication and Authorization Service” on page 403](#)
- [“Addendum A: Setting Typical Categories of Permissions for Project-level Authorization” on page 405](#)

You can implement the authorization manager interface to define three typical categories of user operations such as:

- *Read-only* permission. When users have read-only permission to the repository and project contents they can view entries but they cannot add, delete, or edit them. If they try to add and save a new entry to the directory, they see an error message that tells them that they do not have authorization to add an entry.

- *Read-write* permission. When users have read-write permission to the repository and project contents they can view entries and add, delete, or edit them. When they edit the contents of an entry and save the changes, they will not see any error messages.
- *No-read* permission. When users have no-read permission, they can view the top-level entities but they cannot see their contents in an editor.

Before you can write a custom authorization manager class, you need to decide which users you want to have access to the repository, project, and project contents, where this information is stored, and how the authorization manager will connect to this information. After you have determined how user authentication and authorization needs to be defined, you can implement the `NdRepositoryAuthorizationManager` interface and its methods for repository-level authorization or you can implement the `NdPromAuthorizationManager` interface and its methods for repository and project-level authorization. See “[Writing a Custom Authorization Manager](#)” on page 390.

- If you want to implement repository-level authorization, see “[Handling Repository-level Authorization Requests](#)” on page 392.
- If you want to implement project-level authorization or both repository-level and project-level authorization, see “[Implementing the Blaze Advisor Authorization Manager](#)” on page 391.

If you want to use an authorization manager in an existing repository, after you write your custom authorization manager class, see “[Changing an Existing Repository Configuration](#)” on page 144.

The Authentication and Authorization Example uses a repository configured with a custom authorization manager class. See `<ADVISOR_HOME>/examples/repository/AuthorizationManagerExampleRepository`.

Writing a Custom Authorization Manager

You can write a custom authorization manager by importing the `com.blazesoftware.template.repository.security.NdPromAuthorizationManager`. This interface extends the `com.blazesoftware.repository.base.NdRepositoryAuthorizationManager` interface. You also import other subclasses and interfaces to control what tasks users can perform on entries within the repository.

The `NdPromAuthorizationManager` interface provides access controls to project-level entities based on attributes including SRL type, management property, user credentials, and entry name or path name in your repository, in addition to the repository-level and/or project-level access controls based on the entry name and path name of the entities in the repository.

You can use the `NdPromAuthorizationManager` interface to provide either project-level or repository-level access controls or both.



Note If you have already implemented the `NdRepositoryAuthorizationManager` interface using a prior release of Blaze Advisor, you do not need to update your custom class to continue using it for repository-level access with the current version of Blaze Advisor.

This section contains the following topics:

- “[Adding Import Statements](#)” on page 391
- “[Implementing the Blaze Advisor Authorization Manager](#)” on page 391
- “[Opening and Closing a Repository Connection](#)” on page 391
- “[Handling Repository-level Authorization Requests](#)” on page 392
- “[Handling Project-level Authorization Requests](#)” on page 394
- “[Connection Context and Authorization Requests](#)” on page 400
- “[Setting Authorization Requests for Users](#)” on page 400

Adding Import Statements

When you write your custom authorization manager class, you import the classes and the interfaces with the methods for retrieving information about the repository connection, creating or terminating an instance of the authorization manager and handling authorization requests. The Authentication and Authorization Example uses a repository configured with a custom authorization manager class. See <ADVISOR_HOME>/examples/repository/AuthorizationManagerExampleRepository

Implementing the Blaze Advisor Authorization Manager

You need to implement the `NdRepositoryAuthorizationManager` or `NdPromAuthorizationManager` interface to create an authorization manager instance when a repository connection is opened, terminate a authorization manager instance when a repository connection is closed, and check for permissions when a authorization request is made.

The `NdPromAuthorizationManager` interface contains one method:

```
public void checkPromAuthorizationRequest(NdPromAuthorizationRequest request)  
throws NdPromAuthorizationException;
```

It also uses these `NdRepositoryAuthorizationManager` interface methods:

- `openConnection()` and `closeConnection()`
See “[Opening and Closing a Repository Connection](#)” on page 391.
- `checkAuthorizationRequest(com.blazesoft.repository.base.NdRepository AuthorizationRequest request)`
See “[Handling Repository-level Authorization Requests](#)” on page 392.

Opening and Closing a Repository Connection

The `NdPromAuthorizationManager` inherits from the `NdRepositoryAuthorizationManager` interface which contains two methods: `openConnection()` and `closeConnection()`.

- The `openConnection()` method has the following signature:

```
public void openConnection(NdRepository repository) throws  
NdRepositoryAuthorizationException
```

You use the `openConnection()` method to indicate that a repository connection has been opened and the authorization manager should initialize itself. You should ensure that an exception of type `NdRepositoryAuthorizationException` is thrown if for some reason the authorization manager cannot be initialized.

- The `closeConnection()` method has the following signature:

```
public void closeConnection(NdRepository repository) throws  
NdRepositoryAuthorizationException
```

You use the `closeConnection()` method to terminate the authorization manager instance when the repository connection is closed. You should ensure that an exception of type `NdRepositoryAuthorizationException` is thrown if an authorization manager instance cannot be terminated.

Handling Repository-level Authorization Requests

You implement the `checkAuthorizationRequest()` method of the `NdRepositoryAuthorizationManager` interface to check user permissions for each authorization request made when users want to make changes to directories and items in a repository. This method has the following signature:

```
public void checkAuthorizationRequest(NdRepositoryAuthorizationRequest request)  
throws NdRepositoryAuthorizationException
```

The `checkAuthorizationRequest()` method is invoked each time a user performs an action that requires authorization. If the authorization for the action cannot be granted, an exception of type `NdRepositoryAuthorizationException` must be thrown.

The authorization requests can be triggered for any repository entity. When you specify the permissions at the entry, directory, or item levels, the `NdRepositoryAuthorizationManager` interface processes the authorization requests in the following way:

- Requests that apply to a directory, apply to the contents of the directory only.
- Requests that apply to an entry in the repository, apply to that entry only, which may be either a directory or an item.
- Requests that apply to an item, apply to that item only.

See "[Repository-level Authorization Requests](#)" on page 392.

For more information about repository entities, see "[Repository Entities](#)" on page 24.

Repository-level Authorization Requests

The `checkAuthorizationRequest()` method is invoked whenever the repository needs to trigger the operation described in the `request` argument of type `NdRepositoryAuthorizationRequest`. Each subclass of `NdRepositoryAuthorizationRequest` super class represents a request to perform an operation on a repository entity. The `request` object contains all the information required to navigate up to the repository and its configuration including its connection parameters. We recommend that an instance

of the `NdRepositoryAuthorizationException` is thrown when the request corresponds to an unauthorized operation.

For more information about the methods of the following authorization request classes, see `com.blazesoft.repository.base.NdRepositoryAuthorizationRequest` in the Blaze Advisor API Innovator (Frames version or Non-frames version) in the Blaze Advisor API documentation.

The following of the interfaces, classes and subclasses used for repository-level authorization:

Class	Description
NdRepositoryAuthorizationRequest	Defines the top-most, abstract, repository-level authorization request class that is used for the argument in the <code>NdRepositoryAuthorizationManager.checkAuthorizationRequest()</code> method.
NdRepositoryEntryAuthorizationRequest	Defines the basic authorization for a request on entries in the repository. This is a subclass of the <code>NdRepositoryAuthorizationRequest</code> .
The following subclasses can be used for authorization requests for operations on entries in a repository:	
<ul style="list-style-type: none"> ■ NdRepositoryEntryLockAuthorizationRequest Defines an authorization request to lock an entry in the repository to ensure write access for a given period of time by preventing other users from making any changes. The Lock request is used only by the version manager for the <i>check out</i> function. ■ NdRepositoryEntryUnlockAuthorizationRequest Defines an authorization request to unlock an entry in the repository to ensure that other users are allowed to make changes to the entry. The Unlock request is used only by the version manager for the <i>check in</i> function. 	Defines the basic authorization for a request for entries in a directory. This is a subclass of the <code>NdRepositoryEntryAuthorizationRequest</code> .
The following subclasses can be used for authorization requests for operations on entries in a directory:	

Class	Description
■ NdRepositoryDirectoryEntriesListAuthorizationRequest	Defines an authorization request to list all directory entries within a repository.
■ NdRepositoryDirectoryEntryAddAuthorizationRequest	Defines an authorization request to add a directory entry within a repository.
■ NdRepositoryDirectoryEntryAttributesReadAuthorizationRequest	Defines an authorization request to read directory entry attributes within a repository. An attribute is information about the entry such as its SRL content type.
■ NdRepositoryDirectoryEntryAttributesWriteAuthorizationRequest	Defines an authorization request to read directory entry attributes within a repository.
■ NdRepositoryDirectoryEntryLookupAuthorizationRequest	Defines an authorization request to allow entry lookup within a directory in the repository.
■ NdRepositoryDirectoryEntryRemoveAuthorizationRequest	Defines an authorization request to remove an entry within a directory in the repository.
■ NdRepositoryDirectoryEntryVersionAuthorizationRequest	Defines an authorization request to version an entry within a directory in the repository.
NdRepositoryItemAuthorizationRequest	Defines the basic authorization for a request on an item. This is a subclass of the NdRepositoryEntryAuthorizationRequest.
The following subclasses can be used for authorization requests for operations on items:	
■ NdRepositoryItemContentReadAuthorizationRequest	Defines an authorization request to read contents of an item in the repository.
■ NdRepositoryItemContentWriteAuthorizationRequest	Defines an authorization request to write or edit contents of an item in the repository.

Repository-level Error Messages

The `checkAuthorizationRequest()` method must be declared to throw an `NdRepositoryAuthorizationException`. Any subclass of the `NdRepositoryAuthorizationRequest` should always throw an `NdRepositoryAuthorizationException` to display a message to the user when an authorization request has been denied. We recommend that you provide user-friendly error messages so that users are directed to the cause of the error and what they can do to prevent it in the future. If for some reason the location of the directory or the request is not known, a more generic error message can be displayed.

Handling Project-level Authorization Requests

You implement the `checkPromAuthorizationRequest()` method of the `NdPromAuthorizationManager` interface to check user permissions for each project -level

authorization request made when users want to make changes to items. in a project. This method has the following signature:

```
public void checkPromAuthorizationRequest(NdPromAuthorizationRequest request)  
throws NdPromAuthorizationException;
```

The `checkPromAuthorizationRequest()` method is invoked each time a user performs an action that requires authorization. If the authorization for the action cannot be granted, an exception of type `NdPromAuthorizationException` must be thrown.

The project-level authorization requests can be triggered for any project entity. When you specify the permissions at the directory or item level including its attributes, the `NdPromAuthorizationManager` interface processes the authorization requests in the following way:

- Requests that apply to a project directory, apply to the contents of the directory only.
- Requests that apply to an entry in the project, apply to that entry only, which may be either a directory or an item.
- Requests that apply to an item, apply to that project item only.

See "[Project-level Authorization Requests](#)" on page 395.

For more information about repository entities, see "[Repository Entities](#)" on page 24.

Authorization Request Context

The project-level authorization request context `NdPromAuthorizationRequest` class keeps the authorization request state. Since a single project-level authorization operation is not considered to be atomic meaning that one operation may trigger many other project-level requests and these secondary operations also send authorization requests to the authorization manager. The authorization request context helps the authorization manager identify whether the request is from a primary operation using the `isPrimaryRequest()` method and this value is used for the request state maintained in the authorization context.

The `NdPromAuthorizationRequest.isPrimaryRequest()` method is used in the Authentication and Authorization Example. If you run the example using a command prompt, each time you perform an operation as an authorized user you see the calling sequence where the initial operation triggers other project-level requests. The Authentication and Authorization Example uses a repository configured with a custom authorization manager class. See `<ADVISOR_HOME>/examples/repository/AuthorizationManagerExampleRepository`.

For a description of the project-level authorization requests, see "[Project-level Authorization Requests](#)" on page 395.

Project-level Authorization Requests

The `checkPromAuthorizationRequest()` method is invoked whenever the project needs to trigger the operation described in the `request` argument of type `NdPromAuthorizationRequest`. Each subclass of `NdPromAuthorizationRequest` super class represents a request to perform an operation on a project entity. The `request` object

contains all the information required to navigate up to the repository and its configuration including its connection parameters. We recommend that an instance of the `NdPromAuthorizationException` is thrown when the `request` corresponds to an unauthorized operation.

A request object is constructed based on the operation type and the context entity. For each request check, if a project-level authorization manager is present and the context entities are project-level entries instead of repository-level entries, a corresponding project-level authorization request is constructed and sent to the project-level authorization manager.

For more information about the authorization request classes, see `com.blazesoftware.template.repository.security.NdPromAuthorizationRequest` in the Innovator (Frames version or Non-frames version) in the Blaze Advisor API documentation.

The following of the interfaces, classes and subclasses used for repository-level authorization:

Class	Description
NdPromAuthorizationRequest	Defines the top-most, abstract, project-level authorization request class that is used for the argument in the <code>NdPromAuthorizationManager.checkPromAuthorizationRequest()</code> method.
The following are subclasses of the <code>NdPromAuthorizationRequest</code> class.	

Class	Description
	<p>The following classes define requests at the project level. Users can read the contents of a project, add and remove folders, add and remove subprojects and write the contents of a project.</p> <ul style="list-style-type: none"> ■ NdPromProjectEntryLookupAuthorizationRequest Defines the authorization request for an entry lookup within the project. ■ NdPromProjectDirectoriesListAuthorizationRequest Defines the authorization request to list directories within a project. ■ NdPromProjectDirectoryAddAuthorizationRequest Defines the authorization request to add a directory to a project. ■ NdPromProjectDirectoryRemoveAuthorizationRequest Defines the authorization request to remove a directory from a project. ■ NdPromProjectSubprojectListAuthorizationRequest Defines the authorization request to list subprojects in a project. ■ NdPromProjectSubprojectAddAuthorizationRequest Defines the authorization request to add a subproject to a project. ■ NdPromProjectSubprojectRemoveAuthorizationRequest Defines the authorization request to remove a subproject from a project. ■ NdPromProjectProjectAttributesReadAuthorizationRequest Defines the authorization request to read project attributes. ■ NdPromProjectContentWriteAuthorizationRequest Defines the authorization request to print out the project content. ■ NdPromProjectFilterReadAuthorizationRequest Defines the authorization request to list the filter location. ■ NdPromProjectFilterWriteAuthorizationRequest Defines the authorization request to modify the filter location.
NdPromEntryAuthorizationRequest	Defines the authorization for a request on entries in the project. This is a subclass of the NdPromAuthorizationRequest class.

Class	Description
<p>The following subclasses can be used for authorization requests on entries in a project. The term entry refers to all SRL types that can have management properties and attributes:</p> <p>Users can read and create management properties and they can also edit management property values. They can also read and write the attributes of entities.</p> <ul style="list-style-type: none"> ■ NdPromEntryAttributesReadAuthorizationRequest Defines the authorization request to read project entry attributes. ■ NdPromEntryAttributeWriteAuthorizationRequest Defines the authorization request to write the project entry attributes. ■ NdPromEntryManagementPropertiesListAuthorizationRequest Defines the authorization request to list management properties of the project entry. ■ NdPromEntryManagementPropertiesReadAuthorizationRequest Defines the authorization request to read the management properties of the project entry. ■ NdPromEntryManagementPropertiesWriteAuthorizationRequest Defines the authorization request to write the management properties of the project entry. ■ NdPromEntryManagementPropertyValueReadAuthorizationRequest Defines the authorization request to read a single management property value of a project entry. ■ NdPromEntryManagementPropertyValueWriteAuthorizationRequest Defines the authorization request to write a single management property value of a project entry. 	
NdPromDirectoryAuthorizationRequest	Defines the authorization for a request for entries in a project directory. This is a subclass of the NdPromEntryAuthorizationRequest class.
The following subclasses can be used for authorization requests on entries in a project directory:	

Class	Description
<p>The following classes define requests at the directory level. Users can read the contents of a directory, allow versioning operations, add and remove subfolders, and add and remove entries within a directory.</p> <ul style="list-style-type: none"> ■ NdPromDirectoryEntriesListAuthorizationRequest Defines authorization request for listing entries within a project directory. ■ NdPromDirectoryEntryLookupAuthorizationRequest Defines authorization request for an entry lookup within a project directory. ■ NdPromDirectoryItemEntryAddAuthorizationRequest Defines authorization request for adding an item to a project directory. ■ NdPromDirectoryItemEntryRemoveAuthorizationRequest Defines an authorization request for removing an item from a project directory. ■ NdPromDirectoryEntryCreateAuthorizationRequest Defines an authorization request for creating an entry within a project directory. ■ NdPromDirectoryEntryDeleteAuthorizationRequest Defines authorization request for deleting an entry within a project directory. ■ NdPromDirectoryVersionAuthorizationRequest Defines an authorization request for a versioning operation within a project directory. The request is only sent for the following versioning operations, checkout, checkin, addLabel, removeLabel, and releaseCheckedOutEntries (cancelCheckOut). 	
NdPromItemAuthorizationRequest	
<p>The following subclasses can be used for authorization requests on items:</p>	
<ul style="list-style-type: none"> ■ NdPromItemContentReadAuthorizationRequest Defines an authorization request to read the storage content of a project item. The item storage content is loaded in the NdTemplateManager class. This request is sent only when loading the item storage content. The NdTemplateManager class is the main entry point to the Innovator runtime system. It supports the loading and saving XML documents into a repository. ■ NdPromItemContentWriteAuthorizationRequest Defines an authorization request to write the storage content of a project item into the NdTemplateManager class. 	

Project-level Error Messages

The `checkPromAuthorizationRequest()` method must be declared to throw an `NdPromAuthorizationException`. Any subclass of the `NdPromAuthorizationRequest` should always throw an `NdPromAuthorizationException` to display a message to the user when a project-level authorization request has been denied. We recommend that you provide user-friendly error messages so that users are directed to the cause of the error and what they can do to prevent it in the future. If for some reason the location of the directory or the request is not known, a more generic error message can be displayed.

Connection Context and Authorization Requests

When users connect to a workspace, the workspace connection is made through a Repository Connection Manager that constructs a *repository connection context*. During the construction, if the workspace is configured with an authorization manager, it is examined to see if it is an instance of the project-level authorization manager. If it is, then a project-level authorization manager is set as a private property of the repository connection context.

In a project, wherever a project-level authorization check is required, an associated project-level authorization request is constructed and sent to the project-level authorization manager.

Each operation is checked and if a project-level authorization manager is present and the context entities are project-level entities instead of workspace-level entities, a corresponding project-level authorization request will be constructed and sent to the project-level authorization manager.

Setting Authorization Requests for Users

Before you can write your custom authorization manager class, you need to decide how the authorization manager class will retrieve the logon information so it can authenticate users. For example, you may already have a database or XML file that contains user names and passwords. If you are using a database with your custom class, you need to provide database connection information for the database driver class and the location of the database. If you are using an XML file, you need to provide the path to the file in your custom class. The authorization manager connects to the database table(s) or file(s) to authenticate users and uses the logon information to connect to the repository and check for authorization requests to the directories and their contents.

Setting Root and Directory-level Requests for Repository-level Authorization

The authorization manager you choose, determines the level of authorization available:

- If you implement the `NdRepositoryAuthorizationManager` interface, there are eleven `NdRepositoryAuthorizationRequest` subclasses that you may use to define permissions for the directories where you want to allow users to have access.
- If you implement the `NdPromAuthorizationManager` interface, there are thirty-six `NdPromAuthorizationRequest` classes and subclasses that you may use to define permissions for directories, projects, entries, and items.

Requests set at the most specific directory level take precedence over requests set for the most general level. After a user is authenticated, when they request permission to perform a task, the request is either *allowed*, meaning that the `checkAuthorizationRequest()` method simply returns without failing, or *denied*, meaning that it throws an exception because it cannot have another return state.

For information on repository-level requests, see “[Handling Repository-level Authorization Requests](#)” on page 392.

For information on project-level requests, see “[Handling Project-level Authorization Requests](#)” on page 394.

You can define permissions for individual users or *user roles*. A user role is a way to logically group a set of users who perform similar tasks. For example, if you are setting permissions for the administrator role, you may want to allow anyone with the administrator role to have the ability to read, write, or edit, all directories and their contents. You can specify permissions at the root directory level by using the “/” notation.

If you are setting requests for a business user role for access to directories in a generated RMA, you need to allow read-only permissions at the root directory and provide read-only or read-write permissions to the directories in the generated RMA. For example, if you do not allow a user role read-only permission at the root directory but you allow permission to perform a task such as editing item contents, when users with this role attempt to save their changes, they see an error message.

For information on setting repository-level authorization, see “[Addendum B: Setting Typical Categories of Permissions for Repository-level Authorization](#)” on page 409.

For information on setting project-level authorization, see “[Addendum A: Setting Typical Categories of Permissions for Project-level Authorization](#)” on page 405.

When you refer to the directory, you should use the folder name you see in Windows Explorer. This is what is known as the *storage name*. It is the name that was given to the directory when it was first created in its storage area. If you have changed the name of a directory in the Blaze Advisor IDE, you have changed the *display name* in the Blaze Advisor IDE and not its storage name. This name is used for display purposes and must not be used when specifying permissions to specific directories in the workspace.

General Guidelines When Setting Authorization Requests

The following is a list of general recommendations when setting authorization permissions for user roles at the repository or project level:

- Allow read-only permissions to the root directory, if you are providing access to user roles that will be logging into a deployed rule service.
- Allow the lock and unlock repository-level permissions for directories that are not included in a project.

General Guidelines When Setting Authorization Requests for RMAs

The following is a list of guidelines that apply when setting authorization permissions at the repository or project level for user roles logging into RMAs:

- Allow at least read-only permissions to the root directory even if the user roles that sign into the RMA only require access to the instances.
- Make directory contents inaccessible in an RMA by simply excluding it during the Rule Maintenance Application generation process. Because the directory does not display in the RMA, all users do not have access to it. Only those who have access to

- the Blaze Advisor IDE and the correct permissions to the directory will have access to it.
- Structure your project with directories containing individual instances so that you have flexibility when assigning permissions to user roles. For example, you can exclude/include individual instances in separate folders during the RMA generation process, as well as specify read-only, read-write or partial-write permissions for each instance.

Limitations When Setting Authorization Requests

The following is a list of limitations that apply to both non-versioned and versioned repositories when setting authorization requests for user roles.

You need to set read-only permissions when you want to prevent users from changing entries in a directory. There is currently no way to prevent all users connecting to the same repository from viewing all the directories and their contents in the Repository Explorer. However, you can prevent users from viewing the contents of files in an editor in the Blaze Advisor IDE if you implement project-level authorization and set the permissions to read-only.

Creating a Repository With a Custom Authorization Manager Class

After you have written your authorization manager class and compiled it, you can create the repository that includes the custom class in its configuration. You can use one of the following sets of procedures to add an authorization manager to a repository configuration, see ["Creating a Repository With an Authorization Manager" on page 402](#).

If you have an existing repository where you would like to add an authorization manager, after you write your custom class, you can add it to your repository configuration, see ["Changing an Existing Repository Configuration" on page 144](#).

Creating a Repository With an Authorization Manager

You can create a repository using the New Repository wizard in Blaze Advisor and configure it to use a custom authorization manager class. The creation wizard allows you to add a custom authorization manager class during the configuration process so that you can use an authentication and authorization service with your repository.

To create a repository with an authorization manager

- 1 Select **Repository > New Repository**.
- 2 Enter a name for the repository in the **Repository Nickname** field and use the **Repository Type** drop-down list to select **File Repository (with Authorization manager)**.
- 3 Click the **Details** button located on the lower left side of the wizard.
The Repository Configuration Parameters page opens.

- 4 Click the **Add Authorization Manager** link.
- 5 In the **Manager Class** field, enter the fully qualified name of the custom authorization manager name you want to use.
- 6 Click **OK** to return to the first page of the wizard.
- 7 Click **OK** to exit the wizard.



Note If you made an error while entering your custom authorization manager name after the repository is created, see “[Changing an Existing Repository Configuration](#)” on page 144.

Connecting to a Repository With an Authentication and Authorization Service

After you have created a repository with an authentication and authorization service, you can connect to the repository using the Blaze Advisor IDE, a rule maintenance application (RMA), or a deployed service. In each case, you need to provide a user name and password for authentication and authorization purposes:

- “[Connecting to a Repository With an Authorization Service](#)” on page 403
- “[Connecting to a Repository With an Authorization Manager in an RMA](#)” on page 403
- “[Connecting to a Repository With an Authorization Service in a Deployed Service](#)” on page 404

You can also choose options to save your repository connection credentials in a connection instance, see “[Saving User Name and Passwords](#)” on page 43.

If you are using an authorization manager with non-file BVS repository with private workspaces, you can allow distinct repository and workspace users, see “[Allowing Distinct Repository and Workspace Users](#)” on page 49.

Connecting to a Repository With an Authorization Service

You can open a project stored in a repository configured with an Authentication and Authorization service if connections to the repository are available in the Manage Connections wizard. See “[Connecting to a Blaze Advisor Workspace or Repository](#)” on page 52.

If you need to create a connection to a repository configured with Authentication and Authorization Manager, see “[Connecting to a Blaze Advisor Repository From a Rule Maintenance Application](#)” on page 55.

Connecting to a Repository With an Authorization Manager in an RMA

After you generate a rule maintenance application, if you have used the default generation options, you can enter your user name and password on the Sign In page to connect to the repository. The same custom class you used for the authentication and

authorization service in the Blaze Advisor IDE authenticates users and authorizes requests for adding, editing, and deleting instance files in the RMA.

Connecting to a Repository With an Authorization Service in a Deployed Service

When you deploy a rule service, the user name and password of the user who logged into the Blaze Advisor IDE and deployed the service is added to the server configuration file (.server). If you run the deployed service using the default files that were generated, you do not need to enter the user name and password as command-line arguments because the server configuration file already contains the logon information of a user. However, this also means that the authorization permissions for that specific user will be available to any user who runs the rule service. To change the user name and password, you need to edit the `<JavaMethodArg>` tags nested within the `<InnovatorRepositoryAccessorFactory>` in the `<RulesProjectInnovatorLoaderFactory>` section of the `<RuleServiceAgentFactory>`.

Addendum A: Setting Typical Categories of Permissions for Project-level Authorization

The table below describes typical categories of permissions that you may want to use when working with a file system or any hierarchical set of entities. The project-level authorization request classes listed in this table are in the `com.blazesoftware.repository.template.security` package. For more information about these classes, choose Help > API Reference > Innovator Frames version or Non-frames version).

Permission	Authorization Request Classes	Setting for Behavior
Read-only Users can view the contents of the directories, projects, and items. Users cannot add or edit entries and save them to the repository.		
Directory-level Authorization Requests		
	NdPromDirectoryEntriesList AuthorizationRequest	Allow
	NdPromDirectoryEntryCreate AuthorizationRequest	Deny
	NdPromDirectoryEntryDelete AuthorizationRequest	Deny
	NdPromDirectoryEntryLookupAuthorizationRequest	Allow
	NdPromDirectoryItemAdd AuthorizationRequest	Allow
	NdPromDirectoryItemRemoveAuthorizationRequest	Deny
	NdPromDirectoryVersioning AuthorizationRequest	Deny
Entity level Authorization Requests		
	NdPromEntryAttributesRead AuthorizationRequest	Allow
	NdPromEntryAttributesWrite AuthorizationRequest	Deny
	NdPromEntryManagementPropertiesListAuthorizationRequest	Allow
	NdPromEntryManagementPropertiesReadAuthorizationRequest	Allow

Permission	Authorization Request Classes	Setting for Behavior
	NdPromEntryManagement PropertiesWriteAuthorizationRequest	Deny
	NdPromEntryManagement PropertyValuesReadAuthorizationRequest	Allow
	NdPromEntryManagement PropertyValuesWriteAuthorizationRequest	Deny
	NdPromItemContentRead AuthorizationRequest	Allow
Project-level Authorization Requests		
	NdPromItemContentWrite AuthorizationRequest	Deny
	NdPromProjectContentWrite AuthorizationRequest	Deny
	NdPromProjectDirectoriesList AuthorizationRequest	Allow
	NdPromProjectDirectoryAdd AuthorizationRequest	Deny
	NdPromProjectDirectoryRemove AuthorizationRequest	Deny
	NdPromProjectEntryLookup AuthorizationRequest	Allow
	NdPromProjectProjectAttributes ReadAuthorizationRequest	Allow
	NdPromProjectSubprojectAdd AuthorizationRequest	Deny
	NdPromProjectSubprojectRemove AuthorizationRequest	Deny
	NdPromProjectSubprojectList AuthorizationRequest	Allow
	NdPromProjectFilterRead AuthorizationRequest	Allow
	NdPromProjectFilterWrite AuthorizationRequest	Deny

Permission	Authorization Request Classes	Setting for Behavior
Read-Write Users can view the contents of the directories, projects, and items. Users can add or edit entries and save them to the repository.		
Directory-level Authorization Requests	NdPromDirectoryEntriesList AuthorizationRequest	Allow
	NdPromDirectoryEntryCreate AuthorizationRequest	Allow
	NdPromDirectoryEntryDelete AuthorizationRequest	Allow
	NdPromDirectoryEntryLookup AuthorizationRequest	Allow
	NdPromDirectoryItemAdd AuthorizationRequest	Allow
	NdPromDirectoryItemRemove AuthorizationRequest	Allow
	NdPromDirectoryVersioning AuthorizationRequest	Allow
Entity level Authorization Requests		
	NdPromEntryAttributesRead AuthorizationRequest	Allow
	NdPromEntryAttributesWrite AuthorizationRequest	Allow
	NdPromEntryManagementPropertiesListAuthorizationRequest	Allow
	NdPromEntryManagementPropertiesReadAuthorizationRequest	Allow
	NdPromEntryManagementPropertiesWriteAuthorizationRequest	Allow
	NdPromEntryManagementPropertyValuesReadAuthorizationRequest	Allow
	NdPromEntryManagementPropertyValuesWriteAuthorizationRequest	Allow

Permission	Authorization Request Classes	Setting for Behavior
	NdPromItemContentReadAuthorizationRequest	Allow
Project-level Authorization Requests		
	NdPromItemContentWriteAuthorizationRequest	Allow
	NdPromProjectContentWriteAuthorizationRequest	Allow
	NdPromProjectDirectoriesListAuthorizationRequest	Allow
	NdPromProjectDirectoryAddAuthorizationRequest	Allow
	NdPromProjectDirectoryRemoveAuthorizationRequest	Allow
	NdPromProjectEntryLookupAuthorizationRequest	Allow
	NdPromProjectProjectAttributesReadAuthorizationRequest	Allow
	NdPromProjectSubprojectAddAuthorizationRequest	Allow
	NdPromProjectSubprojectRemoveAuthorizationRequest	Allow
	NdPromProjectSubprojectListAuthorizationRequest	Allow
	NdPromProjectFilterReadAuthorizationRequest	Allow
	NdPromProjectFilterWriteAuthorizationRequest	Allow

Addendum B: Setting Typical Categories of Permissions for Repository-level Authorization

The table below describes typical categories of permissions that you may want to use when working with a file system or any hierarchical set of entities. The repository-level authorization request classes listed in this table are in the `com.blazesoft.repository.base` package. For more information about these classes, choose Help > API Reference > Innovator Frames version or Non-frames version).

Permission	Authorization Request Classes	Setting for Behavior
Read-only Users can view the contents of the directories and items. Users cannot add or edit entries and save them to the repository.	NdRepositoryDirectoryEntriesList AuthorizationRequest	Allow
	NdRepositoryDirectoryEntryAdd AuthorizationRequest	Deny
	NdRepositoryDirectoryEntryLookupAuthorizationRequest	Allow
	NdRepositoryDirectoryEntryRemoveAuthorizationRequest	Deny
	NdRepositoryEntryAttributesRead AuthorizationRequest	Allow
	NdRepositoryEntryAttributesWrite AuthorizationRequest	Deny
	NdRepositoryEntryLock AuthorizationRequest	Deny (for versioned repositories) Note The setting for this authorization class has no effect when using a non-versioned repository
	NdRepositoryEntryUnlock AuthorizationRequest	Deny (for versioned repositories) Note The setting for this authorization class has no effect when using a non-versioned repository
	NdRepositoryItemContentRead AuthorizationRequest	Allow
	NdRepositoryItemContentWrite AuthorizationRequest	Deny

Permission	Authorization Request Classes	Setting for Behavior
Read-Write Users can view the contents of the repository, directory directories and items. Users can also add or delete entries and edit existing entries at any level within the repository.	NdRepositoryDirectoryEntriesList AuthorizationRequest	Allow
	NdRepositoryDirectoryEntryAdd AuthorizationRequest	
	NdRepositoryDirectoryEntry LookupAuthorizationRequest	
	NdRepositoryDirectoryEntry RemoveAuthorizationRequest	
	NdRepositoryDirectoryEntry AttributesReadAuthorization Request	
	NdRepositoryDirectoryEntry AttributesWriteAuthorization Request	
	NdRepositoryEntryLock AuthorizationRequest	Allow (for versioned repositories) Note The setting for this authorization class has no effect when using a non-versioned repository
	NdRepositoryEntryUnlock AuthorizationRequest	Allow (for versioned repositories) Note The setting for this authorization class has no effect when using a non-versioned repository
	NdRepositoryItemContentRead AuthorizationRequest	Allow
	NdRepositoryItemContentWrite AuthorizationRequest	
Partial-Write Users can view the contents of the directories and items. Users can edit existing entries but they cannot add new entries or delete existing entries.	NdRepositoryDirectoryEntriesList AuthorizationRequest	Allow
	NdRepositoryDirectoryEntryAdd AuthorizationRequest	Deny

Permission	Authorization Request Classes	Setting for Behavior
	NdRepositoryDirectoryEntry LookupAuthorizationRequest	Allow
	NdRepositoryDirectoryEntry RemoveAuthorizationRequest	Deny
	NdRepositoryEntryAttributesRead AuthorizationRequest	Allow
	NdRepositoryEntryAttributesWrite AuthorizationRequest	
	NdRepositoryEntryLock AuthorizationRequest	Allow (for versioned repositories) Note The setting for this authorization class has no effect when using a non-versioned repository
	NdRepositoryEntryUnLock AuthorizationRequest	Allow (for versioned repositories) Note The setting for this authorization class has no effect when using a non-versioned repository
	NdRepositoryItemContentRead AuthorizationRequest	Allow
	NdRepositoryItemContentWrite AuthorizationRequest	

CHAPTER 12

Implementing Content Protection for Repository Items

Blaze Advisor provides an Intellectual Property (IP) protection framework consisting of a set of interfaces for protecting proprietary information in a repository. When your repository configuration is updated with a security configuration, *privileged users* connecting to your versioned repository can protect specific files from being viewed by non-privileged users. Your privileged users can also remove protection from repository files if it is no longer needed. Although your non-privileged users can not see the proprietary information in repository item files, they are still able to compile and run projects using these protected files.

Before you can implement the IP protection framework interfaces, you need to decide how the authentication checks are performed against an external authentication service mechanism.

After you have determined the external authentication mechanism, you can implement the `NdAuthenticationService` interface and its methods. See ["Writing the IP Protection Classes" on page 415](#).

To use the IP protection framework in an existing repository, see ["Adding an Authentication Service for IP Protection" on page 196](#).

For additional information about these interfaces, use the FICO Blaze Advisor API Reference:

- 1 In the Eclipse workspace, select to **Help > Welcome** and click the **Overview** icon.
- 2 Scroll down to the FICO Blaze Advisor section and click the **Blaze Advisor API Reference** link.
- 3 Minimize the Welcome page and see the Blaze Advisor API Reference displayed in the Editor pane.
- 4 Select the Innovator (Frames Version) or Innovator (Non-frames version) link and select the `com.blazesoftware.repository.security` package.

About the IP Protection Framework

You use the IP protection framework interfaces to implement custom classes that allow privileged users the ability to view protected item contents and set or remove protection on repository entities. Non-privileged users can still access the repository and its

unprotected contents however they can only view limited information on protected entities. They can also still compile and run projects using these protected files.

The IP protection framework consists of two major parts:

- A framework that allows a Blaze Advisor application to authenticate users against an external service and to retrieve privileged users and the necessary encryption keys to encrypt and decrypt repository content.
- Access to the encryption mechanisms used to encrypt content in the repository. These mechanism can only be invoked either by the Blaze Advisor code base or if the user is a privileged user.

Code Access Security

The IP protection framework provides code access security for the Blaze Advisor code base. It protects access to certain functionality in the Blaze Advisor code base such that the functionality can only be invoked based on the privileged status of the user or the context in which the functionality is invoked.

Code Tampering

In order to prevent code tampering, where the classes in a Blaze Advisor jar file are modified or where additional third-party classes are added to a Blaze Advisor jar file in an attempt to access package level fields in an Advisor class, the Blaze Advisor jar files are signed with a secret key.

Limitations of the IP Protection Framework

The IP protection framework provides a form of digital rights management. The aim of the framework is to prevent a casual inspection of the protected entities by non-privileged users while at the same time not placing a usability burden on legitimate uses of the protected entities.

Implementing the IP Protection Framework

Blaze Advisor provides the classes and interfaces that form the IP protection framework. The framework is designed to receive information from whatever authentication service is used by the customer.

- ["Writing the IP Protection Classes" on page 415](#)
- ["Obtaining Privileged Users and Encryption Keys" on page 416](#)
- ["Connecting the Authentication Service to the Repository" on page 418](#)
- ["Encrypting and Decrypting Protected Item Content" on page 419](#)
- ["Generating Signatures" on page 420](#)

Writing the IP Protection Classes

To implement IP protection for your repository, you write a custom authentication service class by creating an implementation of `com.blazesoft.repository.security.NdAuthenticationService` interface.

The `NdAuthenticationService` interface represents an abstraction of an external authentication mechanism and the functionality required by the IP protection framework to interact with it. Implementations of this interface need to provide the encryption keys and the *authoritative* set of privileged identity tokens to the IP protection framework. An authoritative set of privileged identity tokens provides a secure set of tokens for authenticating users. It also needs to provide the signature generation and verification mechanism that the framework can use to detect any tampering with the repository configuration.



Note The Blaze Advisor IP protection framework provides a definition of this interface, but it is the responsibility of the application developer to provide their own implementation.

The `NdAuthenticationService` interface has these methods:

- `public void obtainAuthenticatedIdentity(NdAuthenticatedIdentityHolder holder NdAuthenticationServiceConnection authenticationConnection, NdWorkspaceConnection workspaceConnection)`
See "[Obtaining Authenticated Identity Tokens](#)" on page 415.
- `public byte[] generateSignature(String data)`
See "[Generating a Signature](#)" on page 416.
- `public boolean verifySignature(String data, byte[] signature)`
See "[Verifying a Signature](#)" on page 416.

Obtaining Authenticated Identity Tokens

The `obtainAuthenticatedIdentity()` method is invoked in order to obtain the set of privileged identity tokens and encryption keys used for encrypting the protected content from within the repository. This method has three parameters:

The first parameter is an instance of the `NdAuthenticatedIdentityHolder` and is used to store the privileged identity tokens and encryption keys. See "[Obtaining Privileged Users and Encryption Keys](#)" on page 416.

The second parameter is an instance of the `NdAuthenticationServiceConnection` and may contain a list of privileged identity tokens. If an instance of the `NdAuthenticationServiceConnection` contains these tokens, they should be copied to the `NdAuthenticatedIdentityHolder` instance by the implementation of this method. See "[About Identity Tokens](#)" on page 416.

The third parameter is an instance of the `com.blazesoft.repository.base.NdWorkspaceConnection` which is the super class for any repository implementation and is used to obtain the user name and password of the workspace user via the repository connection extension and its callback mechanism in the Blaze Advisor IDE or an RMA. This means that when opening a connection to a

protected workspace, if the user wants to be recognized as privileged user, the name supplied for the user must be the name recognized by the authentication service in the framework.

About Identity Tokens

The IP protection framework does not treat the identity tokens that are defined in the repository configuration, `com.blazesoft.repository_config.cfg`, as being authoritative. Instead the framework treats the identity tokens retrieved from the `NdAuthenticatedIdentityHolder` instance as being authoritative. See "[Identifying a Privileged User](#)" on page 417.

From a security standpoint, this distinction is significant because it allows the authentication service to be configured either with a set of privileged tokens via the `NdAuthenticationServiceConnection` instance or be responsible itself for defining a set of privileged tokens. The signature verification method,

`NdAuthenticationService.verifySignature()` prevents a user of a DM suite application from granting themselves elevated privileges via the editing of the set of privileged identity tokens.

By delegating the verification of the signature to the authentication service, attacks against the IP protection framework are avoided. Even if a user could provide another implementation of the `NdAuthenticationService` and have it delegate to the originally configured implementation and the identity tokens were modified in the repository configuration file, provided that the code access security techniques are employed by the IP protection framework to protect the encryption keys returned by the originally configured implementation of this interface, the user would not be able to retrieve those keys from the `NdAuthenticatedIdentityHolder` instance returned by the original implementation.

Generating a Signature

The `generateSignature()` method returns a digital signature for the string passed to the method. The IP protection framework uses this method to generate a digital signature. The `NdRomAdminUtil` utility uses this method to embed a signature into the security section and the authorization manager section of the repository configuration file. See "[Embedding a Digital Signature Into a Configuration](#)" on page 157.

Verifying a Signature

The `verifySignature()` method returns a boolean value indicating whether or not the signature is valid. The IP protection framework uses this method to verify that the configuration file has not been altered when a connection is opened to a protected workspace. The `NdRomAdminUtil` utility uses this method to verify the signatures of the repository configuration files. See "[Verifying a Digital Signature](#)" on page 159.

Obtaining Privileged Users and Encryption Keys

The `com.blazesoft.repository.security.NdAuthenticatedIdentityHolder` interface is used to communicate the privileged identity tokens and encryption keys used in the

authentication service to the IP protection framework. Implementations of this interface are provided by the IP protection framework and its methods are used to wrap the platform specific objects representing the authenticated identities and encryption keys into a form that can be used by other parts of the IP protection framework. These are its methods:

- `public populate(Object identity, String[] privilegedIdentityTokens, Object encryptionKeys);`
This is the only method that the authentication service needs to invoke.
See "[Identifying a Privileged User](#)" on page 417.
- `public String[] getPrivilegedIdentityTokens() throws NdRepositoryException;`
This method returns the set of privileged identity tokens that are passed to the `populate()` method. The IP protection framework uses code access security mechanisms to limit access to this method to the Blaze Advisor code base. See "[Code Access Security](#)" on page 414.
- `public Object getIdentity();`
This method returns the identity object that is passed to the `populate()` method.
- `public String[] getAuthenticatedIdentityTokens();`
This method returns the identity tokens that are associated with the identity of the authenticated user.
- `public Object getEncryptionKeys() throws NdRepositoryException;`
This method returns the encryption keys that are passed to the `populate()` method. The IP protection framework uses code access security mechanisms to limit access to this method to the Blaze Advisor code base. See "[Code Access Security](#)" on page 414.

In the repository configuration, you use the `NdAuthenticatedIdentityHolderFactory` to create the appropriate implementation of the `NdAuthenticatedIdentityHolder`. See "[Instantiating an Authenticated Identity Holder](#)" on page 418.

Identifying a Privileged User

The `NdAuthenticationService` implementation invokes the `populate()` method after successfully obtaining authenticated users and encryption keys from an external authentication mechanism. The method takes three parameters:

The first parameter, `Object identity`, is a platform-specific identity object that represents the authenticated user. The object allows the platform-specific code access security mechanisms to retrieve the identity tokens, such as user name, password, and group, that are associated with the authenticated user.

The second parameter, `String[] privilegedIdentityTokens`, that is passed to the `populate()` method is an array of names of identity tokens. A comparison is made between the set of identity tokens associated with the identity object and the names in the array. If any of the names in the array appear in the set of tokens associated with the identity object, it signifies that the authenticated user is a *privileged* user. The privileged user is allowed unrestricted access to the protected contents in the repository and can also designate repository items as protected or not protected. If a user is authenticated

but not privileged, they can only see the contents of the repository items that are not protected but they can compile and run project using protected items.

The third parameter, `Object encryptionKeys`, that is passed to the `populate()` method is a platform-specific object representing encryption keys that is used to encrypt and decrypt protected repository content and these keys are passed to the encrypted content converter. If a privileged user requests to see the contents of a repository file, these keys are used to decrypt the information. See "[Encrypting and Decrypting Protected Item Content](#)" on page 419.

Instantiating an Authenticated Identity Holder

The `NdAuthenticatedIdentityHolderFactory` interface is used to provide a way to reference the platform-specific implementation of the `NdAuthenticatedIdentityHolder`. This interface has one method:

```
public static NdAuthenticatedIdentityHolder createIdentityHolder()
```

Connecting the Authentication Service to the Repository

The `com.blazesoft.repository.security.NdAuthenticatedServiceConnection` interface allows the authentication service in the framework to connect to the repository. This interface can be extended and configured through the repository configuration file if more information is required to configure a particular implementation of the `NdAuthenticationService` interface. The methods of the `NdAuthenticatedServiceConnection` are used to get and set privileged identity tokens that are passed onto the `NdAuthenticatedService` implementation via its `obtainAuthenticatedIdentity()` method. See "[Obtaining Authenticated Identity Tokens](#)" on page 415.

The methods are:

- `public String getPrivilegedIdentityToken(index i);`
- `public String[] getPrivilegedIdentityToken();`
- `public void setPrivilegedIdentityToken(index i, String token);`
- `public void setPrivilegedIdentityToken(String[] token);`

Configuring the Repository

To protect content in a repository, you need to manually add a security section to the `com_blazesoft_repository_config.cfg` after you create the repository. See "[Adding an Authentication Service for IP Protection](#)" on page 196.

The tags correspond to the `NdSecurityConfig` class that extends the `NdAbstractRepositoryConfigItem` class. The `NdAbstractRepositoryConfigItem` class is the base class for all repository configuration items.

To see the methods of this class in the Blaze Advisor IDE choose `Help > API Reference > com.blazesoft.repository.security` package and scroll down to the classes section to find `NdSecurityConfig`.

The `com.blazesoft.repository.config.NdRepositoryConfig` class which is the super class for all repository configuration classes contains a `getSecurityConfig()` method to

access an instance of the `NdSecurityConfig` class. If the `NdRepositoryConfig` instance (repository configuration file) contains an `NdSecurityConfig` instance, an IP protection framework instance is initialized for the workspace.

Instantiating an Authentication Service

The `NdAuthenticationServiceFactory` interface is used to create an instance of the `NdAuthenticationService` implementation. It contains one method, `createAuthenticationService()` and creates implementations of the authentication service configured from the `NdAuthenticationConnection` instance set on the security section of the repository configuration and passes these instances to the `obtainAuthenticatedIdentity()` method of the authentication service.

Initializing the IP Protection Framework

The IP protection framework is initialized when a workspace is opened containing a repository configuration file where the security section has been added. This initialization invokes the configured authentication server to obtain the privileged user identity and also initializes the code access security mechanisms provided by the framework.

Implementations of the `NdSecurityManager` interface are responsible for installing and uninstalling the IP protection framework. Implementations of this interface are provided by the IP protection framework. The `NdSecurityManagerFactory` interface is used to provide a way to reference the platform-specific implementation of the `NdSecurityManager` interface. This interface contains one method:

```
public static NdSecurityManager createSecurityManager()
```

The method is used to instantiate the appropriate security manager for the platform on which the execution is taking place.

Encrypting and Decrypting Protected Item Content

When privileged users connect to the workspace or a repository configured with an authentication service in the Blaze Advisor IDE, they can use a Protect command to set the protected attribute, `com.blazesoft.repository.security.protected`, located in the associated `.attbs` file of an repository entity from `false` to `true`. The workspace layer is responsible for invoking the code access security mechanisms provided by the IP protection framework to ensure that the protected content is encrypted and decrypted by appropriately privileged users or under the appropriate context. For example, the framework ensures that the protected entities in the workspace or repository have their content encrypted when protected status is set and decrypted when its contents are requested by a privileged user.

Encrypting and Decrypting Content

You implement an encrypted content converter with `com.blazesoft.repository.security.NdEncryptedContentConverter` interface.

The `NdEncryptedContentConverter` interface has these methods:

- `encrypt(byte[] content, Object encryptionKeys);`

- `decrypt(byte[] content, Object encryptionKeys) throws NdRepositoryException;`

If your repository has protected content the workspace layer will install a repository item content converter that will invoke the `NdEncryptedContentConverter` interface for any repository content that has protected content. The value of the `encryptionKeys` parameter passed to the `encrypt()` and `decrypt()` methods is the value that is returned by `NdAuthenticationIdentityHolder.getEncryptionKeys()`.

-  **Note** The Blaze Advisor IP protection framework provides a definition of this interface, but it is the responsibility of the Decision Management (DM) suite application developer to provide an implementation of this interface.

Instantiating an Encryption Content Converter

The `NdEncryptedContentConverterFactory` interface is used to reference the platform-specific implementation of the `NdEncryptedContentConverter` interface. This interface has one method:

```
public static NdEncryptedContentConverter createContentConverter()
```

This method is used by the IP protection framework to create an instance of the `NdEncryptedContentConverter` interface that is specified in the security section of the repository configuration.

-  **Note** The Blaze Advisor IP protection framework provides a definition of this interface, but it is the responsibility of the Decision Management (DM) suite application developer to provide an implementation of this interface.

Generating Signatures

The IP protection framework also includes the `com.blazesoft.util.NdSignatureGenerator` interface to generate digital signatures to protect the security section or the authorization manager section in the repository configuration from being altered by an unauthorized user. Before you can use the IP protection framework and connect to a repository, a digital signature must be generated and added to the security section and if applicable the authorization manager section of the repository configuration.

These are the methods:

- `public byte[] generateSignature(String content, Object privateRSAKey);`
- `public boolean verifySignature(String content, byte[] signature, Object publicRSAKey);`

The types of the private and public RSA keys passed to the methods of this interface are platform-specific.

Instantiating a Signature Generator

The `com.blazesoft.util.NdSignatureGeneratorFactory` class is used to provide a way to reference the platform-specific implementation of the `NdSignatureGenerator` interface. This interface has one method:

```
public static NdSignatureGenerator createSignatureGenerator()
```

This method instantiates the appropriate signature generator for the platform on which the execution is taking place.

The `NdRomAdminUtil` utility is used to embed a digital signature into the security section and/or an authorization manager section of a repository configuration. For the purposes of signing these sections, the authentication service is instantiated. See “[Embedding a Digital Signature Into a Configuration](#)” on page 157.

You can also use the `NdRomAdminUtil` utility to verify the validity of a digital signature, see “[Verifying a Digital Signature](#)” on page 159.

Index

A

Admin Repository
 assigning a management property to an entity category 127
 configuration instances 133
 configuration templates 133
 connecting to 39
 connecting to the Local Admin Repository 132
 connection instances 134
 connection templates 133
 contents 133
 creating a Database workspace connection instance 136
 creating a management property template 124
 creating a management property type 128
 deleting a management property 126
 editing a management property 126
 editing configurations and connections 134
 editing connection instances 134, 136
 editing connection templates 135
 managing configuration and connection instances 131
 manually creating a copy of a connection instance 136
 manually deleting a connection instance 137
 overview of management property templates 123
 system folder 38
 system folder contents 120
 viewing the system folder 119
 writing a connection file 340

administering
 a repository 24
 changing a workspace configuration to add a service 144
 committing a configuration 150
 managing locked files 95
 obtaining a list of locked files 95
 repositories 65
 synchronizing a system folder 137
 unlocking files by user 96
 updating a configuration 151
 viewing the system folder in a repository 119
advanced display options
 list for management properties 114
 management properties 113
allow distinct repository and workspace users
 configuring repositories 49
allow distinct repository and workspace users using the utility 184
assemblies
 for self-contained configuration 155
assembly loader
 .NET 152
AssemblyConfig
 tags for locating an assembly 152
authentication and authorization
 project-level 405
 repository-level 409
authentication and authorization manager. See authorization manager
authentication management
 overview 19
authorization management
 overview 20

authorization manager
adding to a repository 39
connecting to a repository 403
connecting to a repository in a deployed service 404
connecting to a repository in an RMA 403
creating a repository configured with 402
creating a repository in Builder with 402
error messages 394, 399
general guidelines when setting requests 401
handling authorization requests 392, 394
implementing in a repository 389
limitations when setting authorization permissions 402
opening and closing connections to the repository 391
recommendations for setting authorization permissions for RMAs 401
request classes 392, 395
setting authorization permissions 400
setting root and directory-level permissions 400

B

Blaze Advisor Plugin
adding ClearCase jars 264
Blaze Advisor projects
publishing with the utility 79
releasing with the utility 76
replacing published projects using the utility 82
updating published projects using the utility 81

BVS repositories
characteristics of
converting from file repositories 232
creating a BVS File repository with a private workspace in the IDE 217
creating a BVS File repository with a shared workspace in the IDE 219
creating a File (BVS) repository with a shared workspace using the utility 226
creating a File BVS repository with a private workspace using the utility 221
creating in the Blaze Advisor IDE 216
deciding on the workspace type 209
implementing 209
overview to creating a File (BVS) repository using the utility 220
private workspaces 211
shared workspaces 213
structure for a private workspace 212
structure for a shared workspace 214
verifying versioning commands in 230
writing a connection file 175, 221
BVS versioning
updating a workspace with the utility 91

C

categories of project items
management properties 109
class loader
Java 152
ClasspathConfig
tag for locating a custom class 152
ClearCase jars
ClearCase repositories 264

- ClearCase repositories
 - changing the history view 279
 - creating 263
 - creating a view 265
 - creating a view for each user 267
 - creating a VOB 265
 - creating connections to 275
 - creating in Builder 267
 - creating using a branch 270
 - creating using the utility 270
 - creating workspace connections 275
 - editing the repository configuration file 276
 - generating an RMA with a workspaces parameter 277
 - implementing 253
 - jar files needed 264
 - prerequisites for 264
 - sample repository configuration file 206
 - views and Blaze Advisor workspaces 267
- configuration tags
 - for self-contained files 155
 - optional tags for locating custom classes 154
- configuring repositories 184
- connecting
 - to a database repository 52
 - to deployed services 56
 - to existing repositories 52
 - to rule maintenance applications 55
 - to rule services 56
 - to the Local Admin Repository 132
 - with existing connections 52
 - with repositories 52
- connecting to a CVS repository 52
- connecting to a file repository 52
- connecting to a repository
 - authorization service in a deployed service 404
 - authorization service in an RMA 403
- connecting to a repository with an authorization manager 403
- Connection 67, 70
- creating 226
 - File CVS repositories 239
 - management properties 110
 - repositories 33
 - repositories in Builder 28
 - Subversion repositories 253
 - workspaces 226
- writing file for repositories 168
- writing file for workspaces 168
- writing files to create 168
- creating a database table manually
 - changing data types for databases 290
- creating a repository with an authorization manager
 - using Builder 402
- creating a table
 - creating a database table and adding indices for performance 290
 - database repository 287
 - for a database repository 287
 - manually creating a database table 289
 - utility for creating a database table 287
- Creation Date management property 106
- Current Version management property 106
- custom classes
 - loading 152
 - optional tags for locating 154
- CVS repositories
 - characteristics of
 - verifying versioning commands in 230
- CVS versioning
 - updating a workspace with the utility 91

D

- database repositories
 - characteristics 15
 - connecting using an external connection class 306
 - converting to and from a repository 307
 - creating 285
 - creating a BVS Private 293
 - creating a non-versioned 291
 - creating a table 287
 - creating connections 305
 - creating in the IDE 291
 - creating new workspaces 305
 - creating using the utility 296
 - implementing 285
 - setting your classpath 286
 - writing a connection file 173
- Database workspaces
 - writing a connection file 187
- defined
 - repositories 13
- Designing 13

E

- EclipseDropins
 - adding ClearCase jar files 264
- editing
 - management properties 104

entity categories
in repositories 24
exporting
command-line utility arguments for exporting projects 69
editing a generated connection file 63
projects as Advisor projects using a utility 69
repository connections 62
with NdImporterExporterUtil utility 69

F

File CVS repositories 235
access permissions 238
converting a repository to 247
creating 239
creating a branch 237
creating a directory on a CVS server 236
creating a module on the root directory 237
creating a repository directory 236
creating in Builder 239
creating the directory location 236
creating using the utility 242
creating workspace connection in an RMA 246
creating workspace connections 246
creating workspace connections in Builder 246
password-authenticating servers 237
pserver method 237
setting up a server 236
supported versions 236
verifying server installation 236
workspaces 238
writing a connection file 177
File CVS workspaces 238
file repositories
characteristics 15
File repository
writing a connection file 172, 341, 346, 351, 356, 361
File workspaces
writing a connection file 186
filters
query library 122

G

generated configuration files
repositories 38
Generic SCM repositories
Generic SCM repositories. See also SCM repositories

H

Help in the NdRomAdminUtil utility 165

I

Implementing 27, 209, 235, 253, 285, 413
implementing repositories 25
importing
command-line utility arguments for importing Advisor projects 66
projects from Advisor projects using a utility 66
with NdImporterExporterUtil utility 66
innovator_attbs 130, 131
IP protection framework
authentication services 419
code access security 414
code tampering 414
configuring the repository 418
connecting the repository to the authentication service 418
encrypting and decrypting item content 419
generating a signature 416
generating signatures 420
identifying a privileged user 417
implementing in a repository 413
initializing 419
instantiating a signature generator 420
instantiating an authenticated identity holder 418
instantiating an encryption converter 420
limitations 414
obtaining authenticated identities 415
obtaining privileged identities and encryption keys 415
obtaining privileged users and encryption keys 416
overview 413
securing authoritative identities 416
verifying a signature 416
writing the classes 415
Is Versioned management property 106

J

JDBC repositories
setting your classpath 286

L

Last Modified Date management property 106
Last Modified User management property 106

M

management properties
 adding 101, 102
 advanced display options listed 114
 assigning a management property to a category in the Admin Repository 127
categories of project items 109
creating 110
creating a value list 116
creating in the Admin Repository 123, 124
Creation Date 106
Current Version 106
cutting, copying, and pasting items with 118
deleting in the Admin Repository 126
displaying in contents tab 117
displaying in folder editor 117
displaying in project editor 117
editing 104
editing in the Admin Repository 126
Is Versioned 106
Last Modified Date 106
Last Modified User 106
library in the system folder 120
Management Property Names Provider 129
Management Property Type Provider 129
Management Property Values Provider 130
predefined management properties 105
providers 129
providers used for the pre-defined property types 128
removing from contents tab 118
removing from folder editor 118
removing from project editor 118
Repository Content Type 107
Repository Entry Information Value Provider 130
Repository Subtype 107
Repository Target 107
Repository Type 107
setting advanced display options 113
Source Reference 107
template overview 123
Test Case 108
Test Role 108
Version Status 109
viewing in an editor 103
Working Copy Version 109
management property types
 creating in the Admin Repository 128
 overview 128
migrating
 clearCase 7.1 repository 280

MyAssemblyConfig

 tags for locating an assembly 155
MyClasspathConfig
 tag for locating a custom class 155

N

NdlImporterExporterUtil
 exporting projects as Advisor projects 69
 importing pre-6.0 projects 66
NdRomAdminUtil utility
 about 161
 argument values for directory locations 167
 argument values for symbolic names or file paths 166
 Help 165
 running 166
 sample connection and configuration files in Blaze Advisor 170
writing a connection file for a repository 171
writing a connection file for the Admin Repository 340
writing connection and configuration files for 169

O

overview of repositories 13

P

permissions
 project-level authorization 405
 repository-level authorization 409
privacy management 21
project items
 adding a management property to 102
 categories for management properties 109
 cutting, copying, and pasting items with management properties 118
 viewing a management property 103
project requirements
 choosing based on SCM requirements 18
 choosing based on security considerations 19
 choosing repository types 15
projects
 publishing with a utility 79
 releasing with the utility 76

providers
Management Property Names Provider 129
Management Property Type Provider 129
Management Property Values Provider 130
Repository Entry Information Value
Provider 130
used for pre-defined property types 128
published project
replacing with a utility 82
updating with a utility 81
publishing a project
using the utility 79

Q

queries
adding a manager 39
library in the system folder 121

R

releasing a project with a utility 76
removing
repositories 37
workspaces 229, 376
replacing
published projects with a utility 82

- repositories 191
 - about the IP protection framework 413
 - about workspaces 21
 - adding and deleting services 39
 - adding indices for database repository performance 290
 - adding management properties 101
 - adding management properties to an entity category 102
 - Admin Repository 39
 - administering 65
 - administrating 65
 - administration 24
 - BVS repositories 209
 - changing a directory path in connection templates 135
 - changing a workspace configuration to add a service 144
 - changing a workspace configuration to add a sevice 144
 - changing data types for databases 290
 - changing password display for connection templates 135
 - changing to the ClearCase History view 279
 - characteristics of database repositories 15
 - characteristics of file repositories 15
 - choosing a type based on project requirements 14
 - choosing a versioning service
 - choosing based on security considerations 19
 - ClearCase versioning 253
 - committing a configuration 150
 - configuration files 201
 - sample configuration with BVS versioning 303
 - configuration templates 133
 - configuring allow distinct repository and workspace users 49
 - configuring on the details page 39
 - configuring on the details page for saving repository and workspace credentials 43
 - configuring on the details page for saving repository and workspace credentials using the utility 183
 - configuring to allow distinct repository and workspace users using the utility 184
 - connecting in a rule maintenance applications 55
 - connecting to 51
 - connecting to a ClearCase workspace using an RMA 55
 - connecting to a database repository using an external class 306
 - connecting to a deployed service 56
 - connecting to rule services 56
 - connection instances 134
 - connection templates 133
 - connection wizard 51
 - connections 52
 - contents of the Admin Repository 133
 - converting a File repository to a File CVS repository 247
 - converting a repository to a BVS repository 232
 - converting to and from a database repository 307
 - converting to or from another type 50
 - creating 27, 33
 - creating a BVS File repository shared workspace in the IDE 219
 - creating a BVS File repository using the utility 220, 221, 226
 - creating a BVS File repository with private workspaces in the IDE 217
 - creating a BVS repository in Builder 216
 - creating a database BVS Private 293
 - creating a database repository 285
 - creating a Database workspace connection instance 136
 - creating a File CVS repository in Builder 239
 - creating a new connection 58
 - creating a new workspace connection 58
 - creating a repository in Builder 28
 - creating a table for the database repository 290
 - creating a table for the database repository manually 289
 - creating a table for the database repository using a utility 287
 - creating database repositories in the IDE 291
 - creating database repositories using the utility 296
 - creating File CVS repositories using a utility 242
 - database repositories
 - creating in the IDE 291
 - database repositories creating using the utility 296
 - database repository creating non-versioned 291
 - defined 13
 - directories 24
 - editing a generated connection file 63
 - editing a management property 104
 - editing configuration instances 134
 - editing configurations and connections 134

editing connection instances 136
editing connection templates 135
entities in 24
entity categories 24
entries 24
existing connections 52
exporting connections 62
File CVS repositories 235
generated configuration file 38
implementing 25
implementing an authorization manager 389
implementing IP protection framework 413
implementing IP protection framework code
 access security 414
 implementing IP protection framework code
 tampering 414
 implementing IP protection framework con-
 figuring the repository 418
 implementing IP protection framework con-
 necting to the authentication
 service 418
 implementing IP protection framework for en-
 crypting and decrypting item
 content 419
 implementing IP protection framework for
 obtaining privileged identities and en-
 cryption keys 415
 implementing IP protection framework gen-
 erating a signature 416
 implementing IP protection framework gen-
 erating signatures 420
 implementing IP protection framework identi-
 fying a privileged user 417
 implementing IP protection framework
 initializing 419
 implementing IP protection framework in-
 stantiating a signature generator 420
 implementing IP protection framework in-
 stantiating an authenticated identity
 holder 418
 implementing IP protection framework in-
 stantiating an encryption converter 420
 implementing IP protection framework in-
 stantiating authentication services 419
 implementing IP protection framework
 limitations 414
 implementing IP protection framework ob-
 taining authenticated identities 415
 implementing IP protection framework ob-
 taining privileged users and encryption
 keys 416
 implementing IP protection framework secur-
 ing authoritative identities 416
 implementing IP protection framework verify-

 ing a signature 416
implementing IP protection framework writ-
 ing the classes 415
IP configuration 157
items 24
managing locked files 95
managing repository configuration and con-
 nection instances 131
manually creating a copy of connection
 instance 136
manually deleting a connection instance 137
migrating a clearCase 7.1 repository 280
obtaining a list of locked files 95
removing 37
root directory in the Repository Explorer 22
sample configuration with BVS
 versioning 202, 223
sample configuration with CVS
 versioning 204
sample configuration with no services 202
SCM repositories creating in IDE 267
storage names and display names in IDE 23
structure 21, 22
Subversion versioning 253
synchronizing a system folder 137
unlocking files by user 96
updating a configuration 151
use of workspaces 21
verifying versioning commands in 230
viewing a management property in an
 editor 103
writing a configuration file 191
writing a configuration file and adding a
 filter 200
writing a configuration file and adding a query
 manager 199
writing a configuration file and adding a re-
 pository entry filter 197
writing a configuration file and adding a ver-
 sion manager 193
writing a configuration file and adding an au-
 thorization manager 195
writing a configuration file and adding an item
 content converter 198
writing a connection file 171
writing a connection file for a Database
 repository 173
writing a connection file for a File CVS
 repository 177
writing a connection file for a File
 repository 172, 341, 346, 351, 356, 361
writing a connection file for BVS
 repositories 175, 221

writing a custom authorization manager for 390
writing files to create 168
Repository 161
repository
 overview 13
repository administration 65
repository configuration
 optional tags for locating custom classes 154
 sample configuration with BVS
 versioning 223, 303
 sample configuration with CVS
 versioning 204
 sample configuration with no services 202
 sample files 201
 sample repository configuration with BVS
 versioning 202
 schema manager 130
Repository Content Type management
 property 107
Repository Subtype management property 107
Repository Target management property 107
Repository Type management property 107
repository types
 choosing 14
 choosing based on project requirements 15
 choosing based on SCM requirements
 creating a 27
rule maintenance applications
 recommendations for setting authorization manager permissions 401
rule server configuration
 optional tags for locating custom classes 154
running the NdRomAdminUtil utility 166

S

saving repository and workspace credentials
 configuring repositories 43
saving repository and workspace credentials using the utility
 configuring repositories 183
schema manager
 type information 130
SCM
 access permissions 238
 adding a version manager 39
 choosing a versioning service
 CVS password-authenticating servers 237
 verifying CVS server installs 236
SCM repositories
 creating in IDE 267
 deprecated

security controls
 for repositories 19
 writing a custom authorization manager 390
security controls.See **authorization manager**
source control management See **SCM**
Source Reference management property 107
standard business queries
 query library 122
standard queries
 query library 122
storage names
 about 23
 versus display names in IDE 23
Subversion repositories
 creating 253
 creating in Builder 255
 creating using a branch 258
 creating using the utility 258
 creating workspace connection in an RMA 230, 263
 creating workspace connections 230, 263
 creating workspace connections in Builder 230, 263
 implementing 253
 jar files needed 254
 prerequisites for 254
 sample repository configuration file 205, 260
 supported clients and servers 254
system folder
 contents 120
 management properties library 120
 query library 121
 query library builder filters 122
 query library builder queries 122
 query library standard business queries 122
 query library verification queries 122
 viewing in a repository 119
system folders
 in the Admin Repository 38

T

Test Case management property 108
Test Role management property 108

U

updating
 published projects with a utility 81
 updating a workspace with the utility
 BVS versioning 91
 CVS versioning 91

V

value lists
 as management properties 116
Verification Queries folder 122
Version Status management property 109
versioning
 access permissions 238
 adding a manager 39
 CVS password-authenticating servers 237
 CVS pserver method 237
 setting up a CVS server 236
 verifying CVS server installs 236

W

Working Copy Version management
 property 109
workspaces 226
 about 21
 creating a new connection 58
 deciding between a private or shared BVS
 workspace 209
 File CVS local 238
 local workspaces 238
 private 211
 removing 229, 376
 shared 213
 specifying 39
 structure for a private workspace 212
 structure for a shared workspace 214
writing a connection file 185
writing a connection file for a Database
 workspace 187
writing a connection file for a File
 workspace 186
writing file to create 168
writing a configuration file 191
writing a configuration file for a repository
 adding a filter 200
 adding a query manager 199
 adding a repository entry filter 197
 adding a version manager 193
 adding an authorization manager 195
 adding an item content converter 198
writing a connection file for a workspace 185