IBM

# Federated Content Management

## Accessing Content from Disparate Repositories with IBM Content Federation Services and IBM Content Integrator

- Learn about federated content management
- Install and configure Content Federation Services
- Use Content Integrator connectors and client APIs

Wei-Dong Zhu
Roger Bacalzo
Eric Edeen
Yong Jun
Daniel Ouimet
Jason D Schmitt
Bingrong Wang
Daniela Wersin
D Blake Werts
Martin Willingham

Redbooks

Archived

ibm.com/redbooks

Archived

International Technical Support Organization

# Federated Content Management

## Accessing Content from Disparate Repositories with IBM Content Federation Services and IBM Content Integrator

April 2010

Archived

SG24-7742-00

**Note:** Before using this information and the product it supports, read the information in “Notices” on page xi.

Archived

**First Edition (April 2010)**

This edition applies to Version 4, Release 5, Modification 1 of IBM FileNet Content Manager (product number 5724-R81) and Version 8, Release 5, Modification 1 of IBM Content Integrator Enterprise Edition (product number 5724-J31).

**© Copyright International Business Machines Corporation 2010. All rights reserved.**
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

**Notices** . . . . . xi
Trademarks . . . . . xii

**Preface** . . . . . xiii
The team who wrote this book . . . . . xiv
Now you can become a published author, too! . . . . . xvii
Comments welcome . . . . . xvii
Stay connected to IBM Redbooks . . . . . xvii

**Part 1. Introduction** . . . . . 1

**Chapter 1. Federated content management overview** . . . . . 3
1.1 Introduction . . . . . 4
1.2 The case for federated content management . . . . . 5
1.2.1 Businesses need to manage content . . . . . 5
1.2.2 Content . . . . . 6
1.2.3 Content management system . . . . . 7
1.2.4 Enterprise content management . . . . . 7
1.2.5 The silo effect . . . . . 9
1.2.6 The problem: Information silos . . . . . 9
1.2.7 Standardizing on a single platform as a solution. . . . . 10
1.2.8 An alternative solution: Federated content management . . . . . 11
1.3 Federated content management . . . . . 11
1.3.1 Virtual content repository . . . . . 11
1.3.2 Loose coupling . . . . . 12
1.3.3 Typical features . . . . . 12
1.4 Common federation use case scenarios . . . . . 13
1.4.1 Growth by acquisition . . . . . 13
1.4.2 Extending existing ECM capabilities . . . . . 14
1.4.3 Incremental transition to new ECM platform . . . . . 14
1.5 Implementation strategies . . . . . 15
1.6 IBM product offerings for federated content management . . . . . 18
1.6.1 Content Integrator . . . . . 18
1.6.2 Content Federation Services with IBM FileNet Content Manager. . . 19
1.6.3 Integrating or federating content . . . . . 21
1.7 Federated records management . . . . . 22
1.7.1 Records federation services . . . . . 22
1.7.2 Legal discovery . . . . . 24
1.7.3 Records federation or collection . . . . . 25

© Copyright IBM Corp. 2010. All rights reserved. 

1.8 Federated business process management . . . . . 28

**Part 2. IBM Content Federation Services** . . . . . 31

**Chapter 2. Content Federation Services architecture** . . . . . 33
2.1 Master catalog . . . . . 34
2.1.1 Similarities between CFS and Web search engines . . . . . 34
2.1.2 Using an object store as the master catalog . . . . . 35
2.2 Content Federation Services architecture . . . . . 36
2.2.1 Important terms . . . . . 36
2.2.2 High-level architecture . . . . . 38
2.2.3 Content Federation Services operations . . . . . 39
2.2.4 Content Federation Services functional components . . . . . 41
2.2.5 Security of federated documents . . . . . 50

**Chapter 3. Content Federation Services for Image Services** . . . . . 51
3.1 Architecture . . . . . 52
3.1.1 Mapping the Image Services data model to P8 . . . . . 52
3.1.2 Components . . . . . 52
3.2 Federation process . . . . . 55
3.2.1 Federated document operations . . . . . 56
3.3 Planning . . . . . 58
3.3.1 Using Content Federation Services for Image Services . . . . . 59
3.3.2 Your current Image Services system . . . . . 60
3.3.3 Image Services objects to include in federation . . . . . 63
3.3.4 Planning considerations . . . . . 63
3.4 Installation . . . . . 69
3.5 Configuration . . . . . 69
3.5.1 All Windows servers: TCP/IP parameters . . . . . 70
3.5.2 Image Services . . . . . 72
3.5.3 Content Engine . . . . . 80
3.5.4 Testing Content Federation Services for Image Services . . . . . 100
3.5.5 Annotations . . . . . 104
3.5.6 Closed documents . . . . . 108
3.6 Federated records management with Content Federation Services for Image Services . . . . . 112
3.6.1 Lockdown behavior . . . . . 112
3.6.2 Deletion behavior . . . . . 114
3.6.3 Image Services configuration . . . . . 114
3.6.4 Content Engine configuration . . . . . 118
3.6.5 Testing lockdown . . . . . 119
3.7 Troubleshooting . . . . . 121
3.7.1 Log files . . . . . 121
3.7.2 Image Services database table export_log . . . . . 123

3.7.3 Image Services database table doctaba . . . . . 125
3.7.4 Content Federation Services status . . . . . 126
3.7.5 Common errors . . . . . 126
3.8 Tuning . . . . . 130
3.8.1 Bulk import . . . . . 130
3.8.2 Remote caching . . . . . 134
3.8.3 Image Services capacity planning. . . . . 136
3.9 Best practices . . . . . 136
3.9.1 Schema . . . . . 136
3.9.2 Marking sets . . . . . 140

**Chapter 4. Content Federation Services for Content Manager OnDemand**. 145
4.1 Architecture . . . . . 146
4.1.1 Mapping of CMOD data model to P8 . . . . . 146
4.1.2 Components . . . . . 146
4.2 Federation process . . . . . 148
4.2.1 Federated document operations . . . . . 149
4.3 Planning . . . . . 151
4.3.1 When to use Content Federation Services for Content Manager OnDemand . . . . . 151
4.3.2 Best practices . . . . . 152
4.4 Installation and configuration. . . . . 154
4.4.1 Install IBM Content Manager OnDemand . . . . . 155
4.4.2 Install IBM FileNet Content Manager . . . . . 155
4.4.3 Configure IBM Content Manager OnDemand for CFS-CMOD . . . . 155
4.4.4 Configure IBM FileNet Content Manager for CFS-CMOD . . . . . 162
4.5 Federate CMOD documents to FileNet P8 . . . . . 168
4.5.1 Create a custom document class in the Content Engine . . . . . 168
4.5.2 Add properties to the new custom document class. . . . . 170
4.5.3 Map Content Engine properties to CMOD application group fields. 172
4.5.4 Configure and run the CFSOD exporter utility . . . . . 173
4.6 View federated CMOD documents in WorkplaceXT . . . . . 176
4.7 Federated records management with CFS-CMOD . . . . . 179
4.7.1 Records declaration . . . . . 179
4.7.2 Declaring federated CMOD documents as records . . . . . 180
4.7.3 Lockdown behavior . . . . . 181
4.7.4 Deletion behavior . . . . . 182
4.8 Troubleshooting. . . . . 184

**Chapter 5. Content Federation Services implementation using Content Integrator** . . . . . 189
5.1 Architecture . . . . . 190

Archived

5.1.1 Mapping the Content Integrator data model to P8 . . . . . 190
5.1.2 Components . . . . . 191
5.2 Federation process . . . . . 194
5.2.1 Federated document operations . . . . . 194
5.3 Planning . . . . . 197
5.3.1 When to use Content Federation Services implementation (using Content Integrator) . . . . . 198
5.3.2 Planning considerations . . . . . 199
5.4 Installation and configuration. . . . . 202
5.4.1 Deployment architecture . . . . . 202
5.4.2 Configure security on CM8 . . . . . 204
5.4.3 Install RMI proxy connectors. . . . . 208
5.4.4 Configure Connectors in Content Integrator Administration Tool . . 215
5.4.5 Create the CFS federation administrator user . . . . . 224
5.4.6 Install CFS software . . . . . 224
5.4.7 Configure IBM FileNet Content Manager for CFS implementation (using Content Integrator) . . . . . 229
5.5 Federate CM8 documents to FileNet P8 . . . . . 236
5.5.1 Create a CM8 item type to federate . . . . . 237
5.5.2 Create P8 document class to represent CM8 item type . . . . . 240
5.5.3 Create a data mapping between CM8 and P8 . . . . . 243
5.5.4 Create federation rules . . . . . 247
5.5.5 Federate a document . . . . . 252
5.5.6 Verify federated document . . . . . 255
5.5.7 Schedule the federation rule to run automatically . . . . . 257
5.5.8 View federated documents in WorkplaceXT . . . . . 261
5.6 Troubleshooting. . . . . 262
5.6.1 Logs . . . . . 262
5.6.2 View federation rule status . . . . . 267
5.6.3 Installation and Configuration problems . . . . . 271
5.7 Federated records management with CFS (using Content Integrator) . . 275
5.7.1 Lockdown behavior . . . . . 275
5.7.2 Deletion behavior . . . . . 276
5.7.3 Configuring and enabling Enterprise Records . . . . . 277
5.8 Best practices . . . . . 280
5.8.1 Content Integrator RMI proxy connector performance considerations . 280
5.8.2 Exporter performance considerations . . . . . 281
5.8.3 Importer performance considerations . . . . . 283
5.8.4 Configuring CFS implementation (using Content Integrator) for a highly available environment. . . . . 284
5.8.5 Records management-related best practices . . . . . 287

Archived

**Part 3. IBM Content Integrator** . . . . . 289

**Chapter 6. Content Integrator architecture** . . . . . 291
6.1 Architecture overview . . . . . 292
6.2 Integration Services layer . . . . . 293
6.2.1 Connectors and the Connector SPI . . . . . 293
6.2.2 Access services . . . . . 295
6.2.3 Remote Method Invocation (RMI) Proxy Connector . . . . . 296
6.3 Federation Services layer . . . . . 296
6.3.1 Federated search . . . . . 296
6.3.2 Data maps . . . . . 298
6.3.3 Session pools . . . . . 299
6.3.4 Authentication and security . . . . . 300
6.3.5 View services . . . . . 302
6.4 Developer and User Services layer . . . . . 303
6.4.1 Content Integrator API . . . . . 303
6.4.2 SOA Web Services . . . . . 304
6.4.3 Administration Tool . . . . . 305
6.4.4 Common viewer . . . . . 307
6.5 Additional services . . . . . 308
6.5.1 Configuration server . . . . . 309
6.5.2 Logging server . . . . . 309
6.5.3 RMI proxy connector server . . . . . 309
6.5.4 Session pool server . . . . . 312
6.5.5 Single sign-on server . . . . . 313
6.6 General design principles . . . . . 314
6.7 Recent product enhancements . . . . . 315
6.7.1 Product enhancements in Version 8.4 . . . . . 316
6.7.2 Product enhancements in Version 8.5 . . . . . 318

**Chapter 7. Content Integrator installation and configuration** . . . . . 321
7.1 Installing Content Integrator . . . . . 322
7.2 Configuring Content Integrator . . . . . 327
7.3 Configuring Content Integrator connectors . . . . . 329
7.3.1 EMC Documentum connector configuration . . . . . 336
7.3.2 Hummingbird Document Management connector configuration . . . 340
7.3.3 IBM Content Manager connector configuration . . . . . 343
7.3.4 IBM FileNet Content Services connector configuration . . . . . 356
7.3.5 IBM FileNet Content Manager connector configuration . . . . . 361
7.3.6 Microsoft Windows SharePoint connector configuration . . . . . 372
7.3.7 Open Text Livelink connector configuration . . . . . 380

**Chapter 8. Understanding the client APIs** . . . . . 385
8.1 Introduction to the Java API . . . . . 386

Archived

8.1.1 Setting up the environment . . . 387
8.1.2 Javadocs and samples . . . 388
8.1.3 Major Java API classes . . . 389
8.2 A sample client application . . . 391
8.2.1 Basic and secure login . . . 391
8.2.2 Federated and single-repository queries . . . 394
8.2.3 ResultSet operations . . . 398
8.2.4 Retrieval of metadata and native content . . . 401
8.2.5 ItemFinder . . . 405
8.2.6 Locale and time zone support . . . 406
8.2.7 API versus SPI . . . 408
8.2.8 Non-supported operations . . . 409
8.2.9 Performance tips . . . 409
8.3 Advanced Java API topics . . . 412
8.3.1 Session pools . . . 412
8.3.2 Single sign-on . . . 421
8.3.3 Configuration API . . . 426
8.3.4 Federated Query Transformation . . . 429
8.3.5 Security models . . . 430
8.3.6 Complex properties . . . 437
8.4 Web Services . . . 443
8.4.1 Web Services Description Language . . . 444
8.4.2 Services . . . 445
8.4.3 Setting up the environment . . . 446
8.4.4 Samples . . . 447

**Chapter 9. Developing and extending connectors** . . . 449
9.1 Implementing a Content Integrator connector . . . 450
9.1.1 Planning and designing your new connector . . . 450
9.1.2 Implementing your new connector . . . 458
9.1.3 Building, deploying, and configuring your new connector . . . 483
9.2 Extending the existing Content Integrator connector . . . 485
9.2.1 Planning and design . . . 485
9.2.2 Implementing . . . 485
9.2.3 Building, deploying, and configuring your new connector . . . 488
9.3 Connector plug-ins . . . 488
9.3.1 The purpose of a connector plug-in . . . 489
9.3.2 Creating a connector plug-in . . . 489
9.3.3 Packaging a connector plug-in . . . 495
9.3.4 Configuring a connector plug-in . . . 496
9.4 Summary . . . 496

**Chapter 10. Content Integrator industry solutions** . . . 499

Archived

10.1 Direct use of IBM Content Integrator . . . . . 500
10.1.1 Case one: Content access and federated search platform . . . . . 501
10.1.2 Case two: Document exchange platform . . . . . 506
10.1.3 Case three: Federated records solution . . . . . 507
10.2 Embedded use of IBM Content Integrator . . . . . 509
10.2.1 Case one: IBM Omnifind Enterprise Search . . . . . 509
10.2.2 Case two: Licensing and Case management solution . . . . . 510

**Appendix A. Planning and sizing the Content Integrator system** . . . . . 513
Overview and purpose . . . . . 514
Variances applicable to Content Integrator . . . . . 515
Hardware . . . . . 515
Virtualization . . . . . 516
Deployment . . . . . 516
Profile of Content Integrator operations used . . . . . 518
Java version . . . . . 519
Repository speed . . . . . 519
Strategy for system planning . . . . . 519
Determine the load . . . . . 519
Determine deployment configuration . . . . . 520
Determine application usage profile . . . . . 520
Determine desired headroom . . . . . 520
Determine virtualization needs . . . . . 521

**Related publications** . . . . . 523
IBM Redbooks publications . . . . . 523
Other publications . . . . . 523
Online resources . . . . . 523
How to get IBM Redbooks publications . . . . . 524
Help from IBM . . . . . 525

**Index** . . . . . 527

Archived

Archived

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

The following company name appearing in this publication is fictitious:

Car Company A

This name is used for instructional purposes only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in

© Copyright IBM Corp. 2010. All rights reserved.

any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®
DB2®
developerWorks®
Domino®
FileNet®
IBM®
ImagePlus®
Lotus Notes®
Lotus®
Notes®
OmniFind®
OS/390®
Rational®
Redbooks®
Redbooks (logo) ®
System i®
System z®
Tivoli®
WebSphere®
z/OS®

The following terms are trademarks of other companies:

NetApp, and the NetApp logo are trademarks or registered trademarks of NetApp, Inc. in the U.S. and other countries.

FileNet, and the FileNet logo are registered trademarks of FileNet Corporation in the United States, other countries or both.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

JBoss, and the Shadowman logo are trademarks or registered trademarks of Red Hat, Inc. in the U.S. and other countries.

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows NT, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Archived

# Preface

In today's ever-changing environment, it is common for businesses to have valuable operational data spread across multiple content management systems, either through company mergers, acquisitions, or through normal departmental growth. To help discover, manage, and deliver this content, IBM® provides the following products and offerings:

- IBM Content Federation Services
- IBM Content Integrator

This IBM Redbooks® publication introduces the concept of federated content management and describes the installation, configuration, and implementation of these products and offerings.

*IBM FileNet® Content Federation Services*, available through IBM FileNet Content Manager, is a suite of three federated content management services based on the *federation implementation strategy*. Content Federation Services creates a master catalog containing key metadata information for discovered content to be used later for rapid search and retrieval. We describe how to install and configure Content Federation Services for Image Services, Content Manager OnDemand, and IBM Content Integrator. We also address federated records management with Content Federation Services.

Using an *integration implementation strategy*, *IBM Content Integrator* provides a repository neutral API that allows bidirectional, real-time access to a multitude of disparate content management system installations. Unlike Content Federation Services, IBM Content Integrator does not persist any metadata information for discovered content. We present connector configuration details to frequently encountered content management systems. We provide detailed instruction and sample implementations using the product's Java™ and Web Services APIs to access content that is stored in repository systems. We also address advanced topics about creating custom connector implementations, as well as extending the functionality for existing connectors.

This book is intended for IT architects and specialists who are interested in understanding the concerns and strategies around federated content management. Additionally, this book serves as a hands-on technical guide for IT specialists to use when configuring and implementing federated content management solutions.

© Copyright IBM Corp. 2010. All rights reserved.

# The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization.

**Wei-Dong Zhu** (Jackie) is an Enterprise Content Management, Risk and Discovery Project Leader with the International Technical Support Organization. She has more than 10 years of software development experience in accounting, workflow processing, and digital media distribution. Jackie holds a Master of Science degree in Computer Science from the University of Southern California. Jackie joined IBM in 1996. She is a Certified Solution Designer for IBM Content Manager and has managed and led the production of many Enterprise Content Management, Risk and Discovery IBM Redbooks publications.

**Roger Bacalzo** is a Senior Software Developer with the IBM Enterprise Content Management Group in Seattle, WA. He has almost 20 years of experience in enterprise software development with the past four years in IBM/FileNet products. Roger holds a Master of Science degree from the University of Colorado. He is one of the lead developers in the Content Engine 4.*x* project with expertise on the Content Storage and Content Federation Services subsystems.

**Eric Edeen** is an Advisory Software Engineer currently working as a software developer on IBM FileNet P8 Content Federation Services in Kirkland, WA. Eric has over 14 years of experience as a software developer working on IBM Enterprise Content Management products. He holds a Bachelor of Arts degree from the University of Washington. His areas of expertise include Content Storage and Content Federation.

**Yong Jun** is a software engineer with IBM Software Group in Costa Mesa, CA. He has more than 13 years of experience in software development and testing, and he was the test lead for the Content Federation Services 3.5.*x*, 4.0, and 4.5.0 projects. Yong holds a Bachelor of Information and Computer Science from the University of California at Irvine. Yong joined IBM/FileNet in 1999. His areas of expertise include FileNet and IBM FileNet P8 Content Federation Services, and he is an Oracle Certified Professional (Oracle 9i).

**Daniel Ouimet** is an IT Specialist in IBM Enterprise Content Management and has over 20 years of IT experience, including 11 years in Enterprise Content Management. Daniel has held numerous positions, including working for Lab Services as a Technical Consultant, working as the lead architect and technical delivery manager at a large FileNet partner before returning to IBM, and in his current role as an IT Specialist. Daniel holds a Business degree from John Abbott College in Canada.

**Jason D Schmitt** is a Field Delivery Consultant in Austin, TX. He has more than five years of experience in software support and services. Jason holds a Bachelor of Science degree in Computer Science from the University of Texas at Austin. Jason joined IBM in 2003. His areas of expertise include IBM Content Integrator connector development, as well as IBM FileNet products, including Content Federation Services. He is a certified IBM FileNet 4.0 Administrator.

**Bingrong Wang** is a Senior IT Specialist with the IBM Enterprise Content Management Team in San Francisco, CA. She joined IBM Information Management Group in 2001. She has worked with many clients developing enterprise content management solutions. Her expertise includes the enterprise report management solution, compliance and records management solution, content integration, and the content search and discovery solution. She is an IBM certified DB2® Solution Designer, Business Intelligence Solution Designer, and Content Manager Solution Designer. She is also a certified Records Manager Specialist by AIIM. She holds the Master of Business Administration degree from the college of William and Mary and a Bachelor of Engineering degree from the Beijing University of Chemical Technology.

**Daniela Wersin** is a Software Engineer on the IBM Content Integrator software development team in San Jose, CA. She has participated in the design and development of the version 8.3, 8.4, and 8.5 releases. In addition to new feature development, she has helped internal and external clients integrate their applications with Content Integrator. Her areas of expertise include the IBM Content Manager and Content Manager OnDemand connectors and the Java and service-oriented architecture (SOA) interfaces. Daniela holds a Master of Science degree in Business Informatics from the University of Rostock in Germany.

**D Blake Werts** is a Software Engineer and the technical team lead on the IBM Content Integrator software development team in Charlotte, NC. He has over twelve years of experience as both a software engineer and a technical team lead in the Technology and Telecommunications industries. He has expert knowledge in Enterprise Content Management (ECM), Billing, Statement Preparation, and Conditional Access systems, as well as general knowledge of many Customer Relationship Management (CRM) applications. Blake is experienced in the design, implementation, and deployment of client/server and server-side applications, Unified Modeling Language (UML), the IBM Rational® Unified Process (RUP), Relational Database modeling, Distributed Architecture design, and Internationalization/Globalization. Blake graduated from both Clemson University (Computer Science) and Queens University (English Literature).

**Martin Willingham** is a Senior Managing Consultant with IBM Enterprise Content Management Lab Services for IBM US. He has over 20 years of IT experience as an administrator, manager, and delivery consultant. Martin holds a Master of Business Administration degree from Mercer University and a Bachelor of Business Administration degree from the University of the Georgia. Martin joined IBM/FileNet in 2001. His areas of expertise include IBM FileNet Image Services and IBM FileNet P8, data and platform migrations, protected storage, federation, and enterprise systems management. Prior to joining FileNet, he worked with midrange infrastructure, accounting/Enterprise Resource Management (ERP) software, and warehouse management control. He is an IBM FileNet Certified Professional and certified in Six Sigma process improvement methodology.

Special thanks to **Philip Parker** for his contribution of the following article in the Appendix:
“System planning and sizing guidance for IBM Content Integrator”

Thanks also to the following people for their contributions to this project:

Robert Finn
Grace Smith
Christopher Woodrow

William Belknap
Todd Goodykoontz
Richard Hogg
Francis McGovern
Martin Shramo
Bobby White

Sharon Batiste
Barry Beach
Stephane Depuy
Van Do
Donna Dorman
Bob Kreuch
Jenny Lam
Debbie Lelek
Yvonne Santiago
Gabriel Valencia
Alan Voll
Valiant Yiu

Archived

## Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author - all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and client satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

**ibm.com**/redbooks/residencies.html

## Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  **ibm.com**/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/pages/IBM-Redbooks/178023492563?ref=ts

- Follow us on twitter:

  http://twitter.com/ibmredbooks

Archived

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Archived

# Part 1

# Introduction

Archived

© Copyright IBM Corp. 2010. All rights reserved.

Archived

1

# Federated content management overview

In this chapter, we present an overview of federated content management and the types of business problems that federated content management products are intended to address. We introduce the IBM federated content management products. In addition, we briefly address federated records management and federated business process management.

We cover the following topics in this chapter:

- The case for federated content management
- Federated content management
- Common federation use case scenarios
- Implementation strategies
- IBM product offerings for federated content management:
  - Content Integrator
  - Content Federation Services with IBM FileNet Content Manager
- Federated records management
- Federated business process management

© Copyright IBM Corp. 2010. All rights reserved.

## 1.1 Introduction

In today's global economy, the ability to consistently deliver the right information to the right people at the right time can be a potent source of competitive advantage. Companies that are able to do so are those companies that recognize the value of their information assets and make the investments that are necessary to effectively manage them.

According to certain estimates, as much as 80% of the information that a typical business needs to manage is in the form of unstructured content. Examples of unstructured content include scanned images, business forms, Web content, video clips, billing statements, e-mail archives, customer correspondence, and many other information artifacts of various kinds. Effective content management requires the capability to allow content consumers, both internal and external, to search, retrieve, and review content in context, wherever it exists across the entire enterprise.

In order to meet these challenges, businesses have turned to automation to integrate and activate content with business processes. Modern content management systems have assisted considerably in achieving these goals. For many organizations, however, the problem is compounded, because the existing content is distributed across the enterprise in disparate repositories that support separate access methods, functions, and users.

One attractive solution to this problem is federated content management. Federated content management systems provide a common method of interacting with content that is stored in dissimilar repositories. Thus, federated content management allows existing content to remain in place and existing applications to remain intact while at the same time providing a unified solution for managing content across the enterprise.

IBM, the recognized leader in the enterprise content management (ECM) market, offers a complete portfolio of federated content management products that can provide a cost-effective way to transition to ECM without the pain or cost of forced migration.

*IBM Content Integrator* provides a set of flexible services that provides real-time, bidirectional access to disparate content sources through a consistent set of programming interfaces.

*IBM FileNet P8 Content Federation Services (CFS)* is a family of services that provides central control and normalization of enterprise content from multiple disparate content sources through a single, unified enterprise metadata catalog. With CFS, content is made to appear in the IBM FileNet Content Engine as native Content Engine content where it can be retrieved, viewed, used, and

managed, while actually remaining in the existing repository. In addition, it provides the ability for businesses to take advantage of FileNet P8's active content and advanced records management capabilities with federated content.

CFS allows clients to federate content from a variety of sources into Content Engine:

- IBM FileNet Image Services
- IBM Content Manager OnDemand
- IBM Content Manager
- IBM FileNet Content Manager
- IBM FileNet Content Services
- Documentum
- Open Text Livelink Enterprise Server

For the remaining sections of this chapter, we provide a detailed overview of federated content management and the IBM products and services that help companies to manage disparate content using either direct access method (with Content Integrator) or through content federation (with CFS).

## 1.2 The case for federated content management

In this section, we describe the case for federated content management, define it, and explain why it is relevant in the current business environment.

We begin by providing a brief primer on content and content management in general. The primer is intended to provide conceptual and historical context for the discussion about federated content management that follows.

We also describe the silo effect and how it relates to content management. We explain how information silos represent a significant problem for businesses that are attempting to establish a comprehensive enterprise-wide content management strategy. We then identify two possible solutions to this problem: standardizing on a single content management platform and federated content management. The latter of these two solutions is the major focus of this book.

### 1.2.1 Businesses need to manage content

Modern businesses are awash in a sea of content. The ubiquity of sophisticated office productivity tools, especially e-mail, along with advances in the capabilities of digital capture devices, means that new content is being generated at an accelerating pace. At the same time, storage costs have continued to plummet, thus allowing organizations to be much less selective about the data that they

preserve. These technological trends mean that businesses must grapple with the problem of managing ever larger volumes of content data.

Most businesses now embrace the notion that content data, much like more traditional forms of business data, needs to be treated as a strategic asset that requires active management in order to obtain the maximum value from it.

Traditional information management systems have proven to be poorly suited or inadequate for managing content data. It has become evident that effective content management requires specialized tools that are designed specifically for the purpose of managing content data. Technology providers have stepped up to meet this need with the introduction of enterprise-capable content management software products.

Content management, which previously was regarded as a niche market that was served by a few boutique software companies, is now a major segment of the overall market for information management. Content management applications that in the past served the needs of small, highly specialized business processes have matured into mission-critical systems.

### 1.2.2 Content

The word *content* can be ambiguous, so let us define what we mean before we delve further into our discussion of content management and federated content management in particular. Our goal is to define it in terms that are consistent with its usage within the context of content management.

A distinction is often made between structured and unstructured data, with the latter often used as a synonym for content data. *Structured data* is typically associated with traditional database management systems. Database management systems require data to be rigorously defined according to a schema. The *schema* defines the data as collections of records that consist of one or more fields. *Fields* have specific encodings associated with them and represent the data in its elemental form.

The key point about structured data is that it is organized and encoded in a manner that is optimized for search and retrieval. Its structure is tightly bound to the methods and software systems that are used to store and retrieve it. In general, you cannot store any arbitrary data in a database management system without converting it to a form that the database management system understands.[1]

[1] This point is less true today now that modern database management systems are routinely capable of handling large binary objects; however, it was true at the time that the structured as opposed to unstructured terminology was originally introduced.

Content is often referred to as unstructured, which is misleading, because content data is typically extremely structured. However, from the point of view of most content management systems, the internal structure matters little beyond the order of the bytes. This point is significant, because it allows content management systems to treat all content equally, regardless of its internal structure. Thus, content can be any kind of information that can be represented as a sequence of bytes, from a simple text message to a multi-megabyte video clip. And, because no transformation is required to make it usable to the content management system, the internal structure can remain in the form that is best suited to the application that will consume it.

**Clarification:** For the purposes of this book, we will define *content* as an ordered sequence of bytes of arbitrary length and composition.

### 1.2.3 Content management system

In brief, a content management system is a software system that is specialized for dealing with content, such as text, graphics, video, and so forth. Content management system differ from more traditional information management systems by their ability to efficiently handle arbitrarily large data streams of unspecified type.

The architecture of a typical content management system consists of one or more content stores, a searchable catalog, and a set of access services. Content is physically stored in *content stores*, and the *catalog* is used to store metadata. *Metadata* is data that describes a specific item of content and where it is located. Many content management systems support object-oriented APIs that allow specific content items and associated metadata to be managed as a unified whole. These objects are sometimes referred to as *documents*.

### 1.2.4 Enterprise content management

The broad nature of the definition for content management systems means that specific product implementations can vary widely in terms of feature offerings, technical sophistication, and scalability and still have a legitimate claim as a content management product. Clearly, we need greater specificity.

In the early stages of the development of the content management market, individual software vendors tended to focus on a particular market segment, such as Web content management, document management, or work flow. Many early products were also suited only for department-scale applications.

Archived

Over time, a few companies, including IBM, had the resources and technical wherewithal to continue to expand their product feature set to encompass multiple market segments, while simultaneously improving their ability to handle ever larger volumes of data and transactions with greater reliability. As the market matured, industry analysts and technical professionals recognized the need to differentiate those products and companies that supported truly enterprise-level capabilities. Thus, the term enterprise content management (ECM) was coined.

True ECM products, such as the IBM FileNet Content Manager and IBM FileNet Business Process Manager, are characterized by offering a comprehensive suite of content management capabilities combined with the ability to support and sustain enterprise-level operations.

Next, we briefly describe the types of functional capabilities that are typically found in ECM products.

### Document management

The document management component is often regarded as the foundation of an ECM system. This component provides the fundamental document management services:

- Check-in and check-out
- Version control
- Security services
- Search and retrieval

### Records management

*Records management* is the field of management that is responsible for the efficient and systematic control of the creation, receipt, maintenance, use, and disposition of records, including processes for capturing and maintaining evidence of and information about business activities and transactions in the form of records.

### Digital capture

*Digital capture* functionality focuses on a product's capability to acquire content in digital format by scanning documents and capturing faxes. ECM systems often support the high volume conversion of paper documents into an electronic format.

### Collaboration

*Collaboration* refers to services that are designed to make it easier for project teams to produce and consume content as part of a joint effort to achieve a shared goal.

Archived

### Workflow and business process management

The business process management component of ECM integrates content data with the business processes that create and consume it. Business process management functionality includes these components:

- Graphical tools for defining and displaying business process sequences
- Content routing and task assignment
- Work queue management
- Process auditing

### Web content management

Web content management focuses on providing tools for creating and managing content that appears on Web sites. Web content management systems typically include facilities for allowing content specialists, who are not necessarily also Web specialists, the ability to publish to a Web site. Structural and presentation standards are often enforced by the use of templates.

## 1.2.5 The silo effect

Business analysts often refer to an organizational phenomenon that is known as the *silo effect*. The name derives from the tall windowless structures that are used to store grain and other agricultural commodities. The image of many silos arrayed side by side, built together to serve a common purpose, yet separated into distinct units, is a powerful metaphor for a particular type of operational inefficiency that is common in larger organizations.

The silo effect is characterized by a lack of horizontal integration among the subdivisions that exist within a larger organization. In terms of the metaphor, the silos represent the individual business subunits, and the silo walls represent the structural impediments that hinder effective collaboration between them. When businesses suffer from the silo effect, the cooperative exchange of information often occurs freely and efficiently within the confines of the organizational silo, but it is severely restricted or even prevented from occurring beyond its boundaries.

## 1.2.6 The problem: Information silos

The existence of incompatible information systems within the same organization is a major cause of the silo effect. The overall value of the information that is stored and managed by these systems is severely diminished, because it tends to be bound to specific business processes and cannot easily be repurposed to support other business needs.

Archived

When information is segregated in this fashion, the repositories in which it is stored are sometimes referred to as *information silos*. Similar to organizational silos, information silos allow information to flow freely within the context of the owning repository; however, it cannot easily be shared and combined with related information that is contained in other repositories.

**Clarification:** We use the term *repository* throughout this book to mean a logical container for content and the specific software system that manages it.

This problem afflicts content management systems in the same way that it afflicts more traditional information management systems. The adoption of content management systems has tended to proceed in an evolutionary fashion, often without an overall plan to address the needs of the entire enterprise. This process has resulted in a proliferation of multiple content management systems within the same organization. These systems are often departmental in scope and, in many cases, contain content that has been duplicated from other repositories. These other repositories include e-mail systems and network file systems, as well as vendor-supplied or home-grown content management systems. The existence of these disparate systems has created a significant barrier for organizations attempting to achieve their goals of implementing a comprehensive, enterprise-wide, content management strategy.

In addition, early adopters want to take advantage of the newer features and technology available from more recent products; however, doing so often means abandoning substantial investment in their current content management systems.

### 1.2.7 Standardizing on a single platform as a solution

One potential solution to the problem of information silos is to simply eliminate them by standardizing on a specific product platform. This standardization involves migrating content that is stored in existing repositories into a single repository that is supported by the standard platform and, then, retiring the nonstandard repositories. This approach offers significant benefits and can be a viable option for many organizations; however, for other organizations, the high cost of replacing existing technology and the attendant business disruption is prohibitive.

### 1.2.8 An alternative solution: Federated content management

An alternative to consolidating content under a single platform is to use a content management system that supports *federated content management*. Federated content management systems are a type of content management that is specifically designed to address the problems that result from content that has been segregated into information silos. Federated content management systems are the focus of the remainder of this book. Next, we provide a brief overview of federated content management.

## 1.3 Federated content management

In an environment that includes multiple content management systems, discovering and accessing content for use, reuse, or remix tend to be highly idiosyncratic and dependent on the repository type. Critical aspects, such as user authentication, query syntax, schema, and schema discovery, can vary considerably among the separate repository types. In these cases, the ability to share information between repositories is typically extremely limited and often based on solutions that cannot be broadly replicated.

A solution to this problem is *federated content management*. Federated content management refers to a capability of certain ECM systems that addresses the management of content distributed among multiple heterogeneous repositories. By *heterogeneous*, we mean that the repositories differ from one to another in ways that limit opportunities for achieving interoperability. Federated content management systems are designed specifically to address this problem.

Federated content management systems offer an alternative to consolidating content under a single platform that many businesses and organizations find attractive, because it allows them to preserve their investment in their existing content repositories. Federated content management systems allow existing content to stay in place in its current location, and, thus, the content remains transparent to existing applications.

### 1.3.1 Virtual content repository

Federated content management systems typically create what can be viewed as an abstract virtual repository that is backed by one or more actual repositories. The virtual repository provides an enterprise view of all content assets regardless of where they are stored in the enterprise. The virtual repository exposes a repository-neutral API that is implemented as a software layer that sits on top of the native APIs of the participating external repositories. Federated

content management applications are implemented using the common, repository-neutral API.

### 1.3.2 Loose coupling

One of the key characteristics of federated content management systems is implied by the presence of the word *federated* in the name. A political federation connotes a loose coupling of participants with a weak central authority or even no central authority. Political federations tend to achieve common goals by cooperation among the member states according to well defined protocols rather than by completely ceding authority to a higher office. Individual states within the federation maintain autonomy internally while simultaneously presenting a unified outward facing political entity when dealing with non-member states.

The metaphor is apt for federated content management, because federated content collections are typically loose constructs in which participating repositories retain a large measure of autonomy; however, federated content collections present a common interface to content consumers.

The idea of the federated model is to create a mechanism by which content in external repositories can be accessed easily using applications that are not specialized for a specific repository type. Also, the intent is to allow access so that the repository-neutral applications are transparent to native applications.

Federated content management systems typically communicate with external repositories using the repository's native API, so that any existing applications supported by the repository's native API can continue to function without modification. Also, the external repositories continue to *own* the content that they contain, which means that access to the content is still under the direct control of the repository's native services.

### 1.3.3 Typical features

Federated content management systems typically support: single point of access, federated search, federated security, federated records management, and federated business process management.

#### Single point of access

Federated content management systems support uniform access to content and metadata using a common, repository-neutral API. This API allows applications to access content transparently in any participating repository while shielding the applications from the idiosyncrasies of each repository's API. Another advantage of federation is that you have a single application to access all of the data, not

only an API. For instance, in FileNet P8, you can access all of your federated content through WorkplaceXT without having to go to separate user interfaces and applications for each of the repositories.

### Federated search

Federated search is the capability to search multiple repositories simultaneously from a single point of access and to obtain a fully aggregated result set in return. *Aggregation* is the process of combining search results from separate sources in a helpful way, such as sorted by title, for example.

### Federated security

*Federated security* means that the same credential can be used to access all repositories that participate in the federation. After the repositories have been federated, a user only has to sign on to access the federated repository.

### Federated records management

*Federated records management* provides a centralized facility for performing records management operations on federated content even when the underlying repository provides no native support for records management.

### Federated business process management

*Federated business process management* is the capability to support business process management features on federated content. For example, changing a property on a federated document (that indicates that it has been approved) can cause a step in an automated business process to be completed.

## 1.4 Common federation use case scenarios

In this section, we describe several common use case scenarios in which federated content management systems are useful.

### 1.4.1 Growth by acquisition

Organizations often grow by acquiring and merging with other companies. In doing so, they often inherit existing content management repositories, which can result in an immediate onset of the silo effect because there is no longer a common system for managing content. Thus, the full benefit of the acquisition cannot be realized.

Archived

A federated content management system can offer an ideal solution to this problem by providing a single point of access with a common interface for both existing content repositories and those systems obtained as part of the acquisition. This solution allows organizations to quickly obtain the benefit of a unified content management system without needing to engage in a costly and time-consuming data conversion exercise.

### 1.4.2 Extending existing ECM capabilities

Many organizations have made a substantial investment in their existing content management systems, both in custom applications and associated business processes. The ever changing nature of most business environments at times results in the emergence of new content management requirements that the current platform cannot satisfy. Newer and more sophisticated products can offer solutions to these problems; however, conversion is costly and can be highly disruptive.

Federated content management can offer a solution to this problem by providing extended functionality that can be applied to federated content. This approach effectively creates another layer of functionality above the repository's native capabilities. Due to the inherent nature of the federation approach, you can implement federated content management transparently with little or no impact to the underlying repository. Therefore, you preserve the existing investment by allowing current applications to remain intact. For example, IBM Enterprise Records (formerly known as IBM FileNet Records Manager) offers advanced records management capabilities. When used in conjunction with IBM Content Federation Services, you can apply these capabilities to content that is federated from repositories that do not support records management features as part of their native functionality.

### 1.4.3 Incremental transition to new ECM platform

Migration of content from existing repositories to a new ECM platform might be the ultimate goal of an organization; however, in many cases, you cannot perform this migration during a normal system shutdown period. A lengthy transitional phase necessarily results in which both the old system and the new system must operate simultaneously without disrupting existing business processes.

A federated content management system can allow an organization to immediately reap the benefits of the new system during this transitional period without disrupting the existing business processes that are based around the retiring system.

## 1.5 Implementation strategies

Broadly speaking, federated content management solutions tend to adopt one of two implementation strategies: the *integration strategy* and the *federation strategy*.

> **Tip:** Federated content management products can include both implementation strategies. Many organizations choose to employ both types of implementation strategies, depending on the specific use cases that they need to support.

### Integration strategy

Figure 1-1 depicts an abstract model of a federated content management system that is based on the integration implementation strategy.

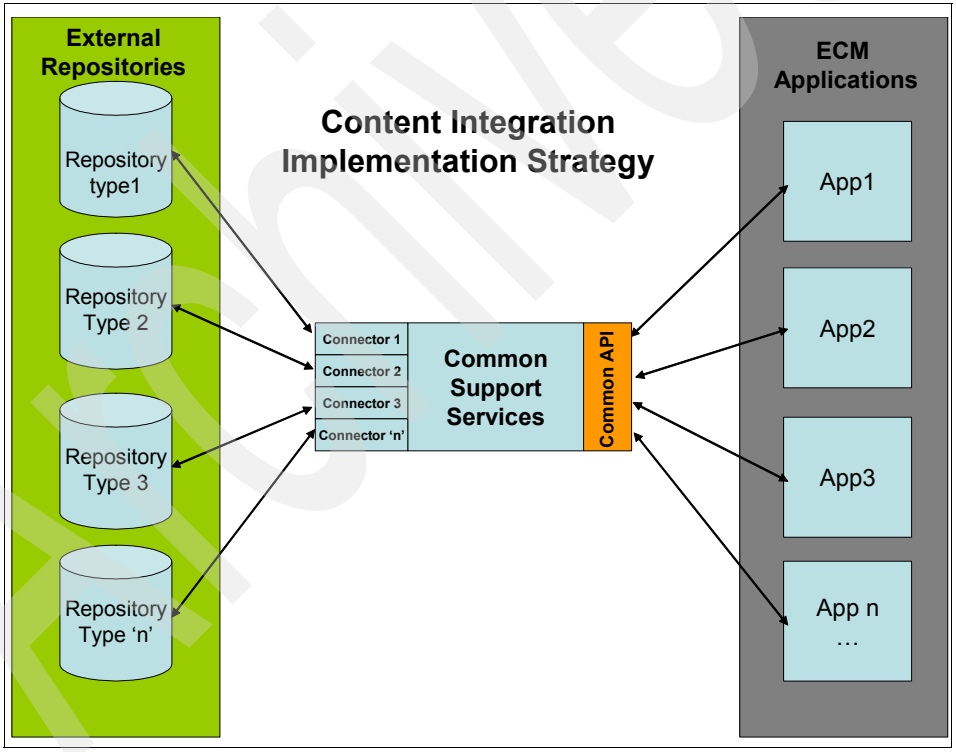



*Figure 1-1 A federated content management system based on the integration strategy*

Archived

Federated content management systems that use the content integration implementation strategy typically provide these types of features:

- Applications access content through an intermediate layer that provides a repository-neutral API.
- All access to content that is stored in external repositories, including updates, occurs in real time.
- Integration-based systems provide a translation layer that converts the common API calls into the equivalent repository-specific calls. The translation layer tends to be repository-specific and requires a separate implementation for each repository type. Often, we refer to these implementations as *connectors*.
- Implementations typically include several non-repository-specific support services that are layered above the connectors, for example:
  - Mapping of native repository schema to a repository-neutral format
  - Query parsing and result set collation
  - Credentials management
  - Session management

### Federation strategy

Figure 1-2 shows an abstract model of a federated content management system that is based on the federation implementation strategy.

Archived

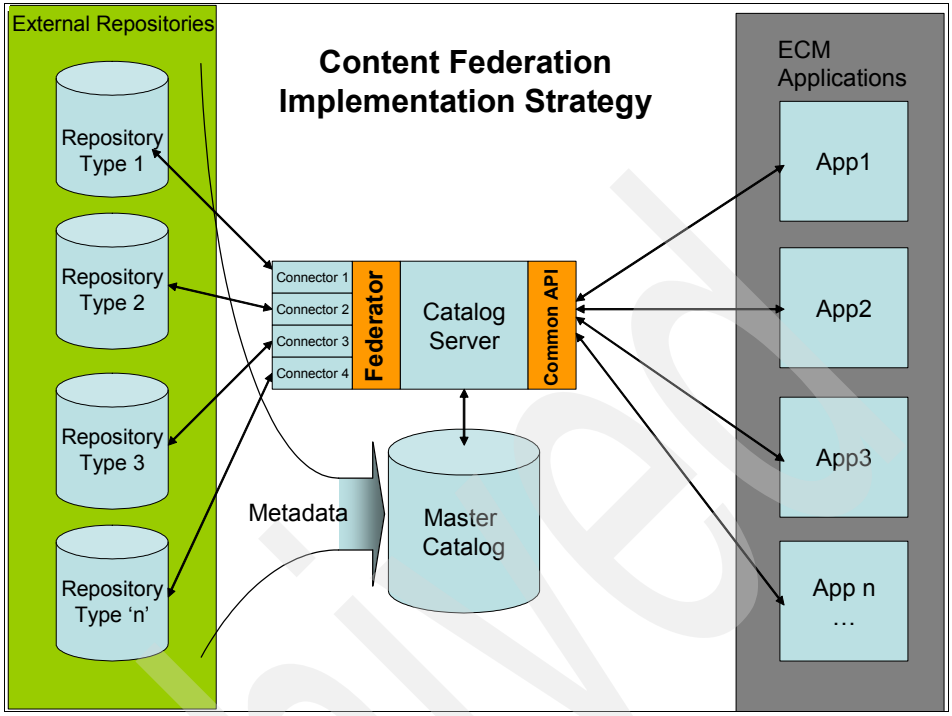



*Figure 1-2 A federated content management system based on the federation strategy*

Federated content management systems that use the content federation strategy share many of the same characteristics and features as those systems that are based on the integration strategy. However, several important differences exist:

- Content metadata from multiple content repository sources is captured and consolidated into a centralized master catalog.
- Distributed content must be cataloged in the master catalog before it can be accessed from the common API.
- Read access to external content occurs in real time, but updates do not.
- Queries execute much faster, because all federated content is referenced from the same catalog.

Archived

## 1.6 IBM product offerings for federated content management

The IBM enterprise content management product suite includes several products that support federated content management features. Content Integrator is based on the integration implementation strategy. IBM FileNet Content Federation Services, which is part of the IBM FileNet Content Manager, is based on the federation implementation strategy.

### 1.6.1 Content Integrator

Content Integrator is a federated content management product that is based on the integration implementation strategy. It provides a repository-neutral API that allows bidirectional, real-time access to many content repository types.

The repository-neutral API allows users to perform many operations, such as:

- Search and retrieval.
- Create, check in, and check out content.
- Create and manage folder hierarchies.
- View and update metadata.
- Retrieve native content for viewing.
- Discover a schema definition.
- View and update security.
- Workflow operations.

Content Integrator's federated search capability enables property-based and full text queries to execute across multiple repositories that return aggregated result sets. It also supports a directed search capability that allows a query to execute against a specific repository.

Content Integrator provides a schema mapping facility that allows administrators to create a normalized view of metadata for multiple repositories with dissimilar metadata schemas.

Content Integrator offers both a Java-based API and a service-oriented architecture (SOA) Web services interface. It is designed to be lightweight so that you can embed it with other products, such as IBM Content Federation Services. Also, you can install and use it independently to provide a consistent interface for multiple heterogeneous content repositories.

We describe Content Integrator fully in Part 3, "IBM Content Integrator" on page 289.

Archived

## 1.6.2 Content Federation Services with IBM FileNet Content Manager

IBM FileNet P8 Content Federation Services is actually a suite of three federated content management services. These services are available with IBM FileNet Content Manager (the product) and are all based on the federation implementation strategy. These services include: Content Federation Services for Image Services (CFS-IS), Content Federation Services for IBM Content Manager OnDemand (CFS-CMOD), and CFS implementation (using Content Integrator).

All three CFS services share the following features:

- These services fully integrate with the FileNet P8 Platform.
- Content metadata is captured and cataloged into a designated IBM FileNet object store, which operates as the master catalog.
- Actual content remains in the external repository; however, it is fully accessible using the P8 Content Engine API.
- Integrated search capabilities allow content to be discovered in any participating external repository.
- Federated content can be seamlessly integrated with non-federated content.
- The full suite of IBM FileNet applications, including those applications that are offered by IBM Business Partners, can operate transparently on federated content without modification
- You can use Enterprise Records to enforce records management policies on content that is federated from external repositories.
- Applications that are based on the external repository's native API can continue to function without modification.
- These services support event driven workflow on federated content using IBM FileNet Business Process Manager.

### Content Federation Services for Image Services

*IBM FileNet Content Federation Services for Image Services (CFS-IS)* is used to federate content from IBM FileNet Image Services repositories, which is the IBM industry-leading image services platform. Image Services (IS), also referred to as *Image Manager*, is a high-performance and highly scalable imaging solution that leverages advanced caching and a distributed architecture to manage large volumes of critical business information.

Currently, thousands of Image Services implementations exist worldwide, making CFS-IS an extremely important and popular offering. CFS-IS allows you to preserve the investment that is made in Image Services while capitalizing on the

Archived

breadth of functionality that is available in IBM FileNet, such as records management.

CFS-IS allows businesses to use high speed ingestion methods, such as High Performance Image Import (HPII), IBM FileNet Capture, or other Image Services (IS) ingestion tools. Content that is captured and stored in Image Services is cataloged in P8 using CFS-IS.

With CFS-IS, content that is stored in Image Services can be leveraged by IBM FileNet Business Process Manager and Enterprise Records and made accessible through federated search.

### CFS for Content Manager OnDemand

*IBM FileNet Content Federation Services for Content Manager OnDemand (CFS-CMOD)* is used to federate content from IBM Content Manager OnDemand. Content Manager OnDemand is an automated archival and retrieval system that stores printed output, such as reports, statements, invoices, and image documents. It processes printed material and stores it on various types of storage media, including hard disks, optical platters, and tapes. The system extracts index values from the printed output and stores those values in a relational database, such as DB2. After the printed output and index values are stored, you can use any of the OnDemand client programs to query, retrieve, and view the documents.

CFS-CMOD exposes content stored in an IBM Content Manager OnDemand repository to a broad range of FileNet P8 applications, including Enterprise Records and IBM FileNet Business Process Manager.

### CFS implementation (using Content Integrator)

CFS implementation (using Content Integrator) is an implementation of CFS that is based on Content Integrator technology. CFS implementation (using Content Integrator) provides connectivity and schema mapping services to allow content from the following repositories to be federated to IBM FileNet Content Manager:

- IBM FileNet Content Services
- IBM FileNet Content Manager
- IBM Content Manager
- Documentum Content Server
- Open Text Livelink Enterprise Server

In addition to these repository types, IBM Lab Services and IBM Global Services also offer services for developing custom third-party repository connectors in order to extend the reach of CFS to embrace additional repositories. For more information about these services, contact your local sales team.

We describe Content Federation Services fully in Part 2, “IBM Content Federation Services” on page 31.

### 1.6.3 Integrating or federating content

Deciding whether to integrate content or federate content determines which IBM federated content management product is best suited to the needs of your organization. The answer depends partly on your currently installed base of content repositories and partly on the use cases that you need to support. It is also important to note that these products are not mutually exclusive. You might find that a blend of IBM federated content product offerings provides the best possible solution for your business needs.

Choose Content Integrator in these cases:

- You need to support use cases that require real-time updates to content in external repositories.
- You need to federate content from a repository type that is not currently supported by any of the CFS products.

Choose Content Federation Services in these cases:

- Your existing repository mix includes one or more of the following repository types:
  - IBM Content Manager
  - IBM Content Manager OnDemand
  - IBM FileNet Content Manager
  - IBM FileNet Image Services
  - IBM FileNet Content Services
  - Open Text Livelink Enterprise Server
  - Documentum Content Server
- You require high performance cross-repository search capability.
- You want to use Enterprise Records (formerly known as IBM FileNet Records Manager) on federated content.
- You want to use IBM FileNet Business Process Manager on federated content.
- You plan to eventually make FileNet P8 Platform your standard content management platform.

Archived

## 1.7 Federated records management

In order to meet current and future regulatory and compliance challenges, organizations need to find ways to effectively manage their business records through their life cycles. Having multiple independent silos of information across business units makes it difficult to insure compliance and creates problems in meeting discovery requirements, which can lead to legal penalties.

An effective method for streamlining an organization's records management efforts is to federate content from multiple source repositories to a single IBM FileNet Content Manager repository.

### 1.7.1 Records federation services

Records federation services refers to the ability to records-enable source repositories, making it possible to manage the content in those repositories with all phases of the record's life cycle. Records federation services provides capabilities for record declaration, classification, reviews, holds, audits, reporting, security, access, transfers, exports, and destruction.

*IBM Records Federation Services* is a product that delivers a multiple repository solution to help your organization manage all of its records through one centralized records management application, Enterprise Records, with the use of Content Federation Services. Records control is applied directly to documents within the business applications' repository. Records Federation Services leaves records in their native repositories, keeps business processes intact, preserves vital security, and unburdens the user from records management overhead. Figure 1-3 on page 23 is a high-level diagram of Records Federation Services.

Archived

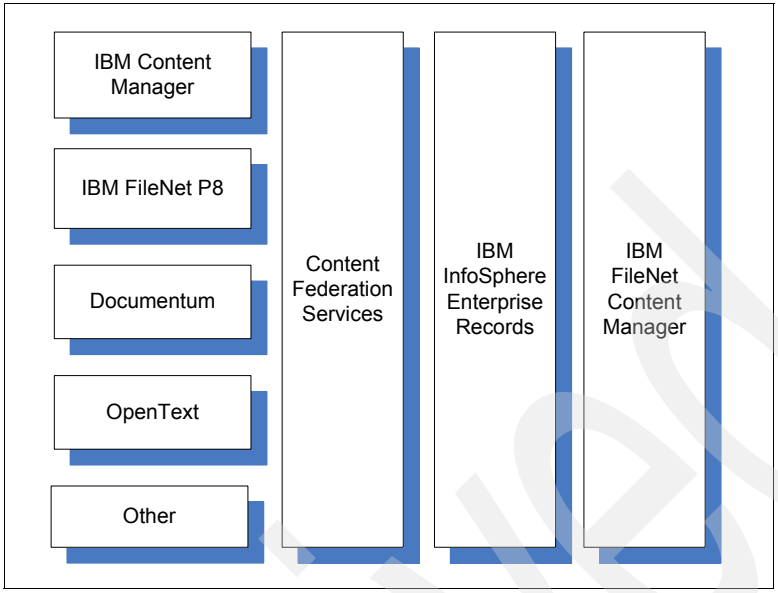



*Figure 1-3 Records Federation Services*

A key advantage of this approach is the ability to utilize a single file plan and retention schedule for all federated content, thus, greatly simplifying the administration of corporate records.

A document federated to FileNet P8 can be declared as a record, which starts a chain of records management events. The record declaration assures that the source document is locked and assigns the source document a record policy. This policy dictates the rules for retention and disposition.

An integral part of the records management process is ensuring that documents that have been declared as records are deleted according to the records disposition schedule. After a document is declared as a record, it is automatically locked down so that the content of the existing version can only be deleted by authorized users with the records management application.

*Lockdown* is the action of enabling protection on the content in order to prevent any changes to it or its deletion. Records Federation Services leverages the source repository's API to lock down the source content; therefore, the methods for lockdown can vary.

See *Understanding IBM FileNet Records Manager*, SG24-7623, for more details about Enterprise Records (previously known as IBM FileNet Records Manager):

http://www.redbooks.ibm.com/abstracts/sg247623.html?Open

## 1.7.2 Legal discovery

Having multiple information silos makes it difficult to locate documents during legal discovery. In addition, having multiple silos makes it impossible to insure that you have the correct version of a document, because several versions or copies of a document can exist in one or many repositories. This situation is a liability in situations where content that must be destroyed (in accordance with the organization's retention policies) has not been destroyed. That content is now discoverable.

When federating the content from these disparate repositories, you can help to mitigate the risks by using Enterprise Records. Content that is federated to IBM FileNet content repository can be declared as a corporate record using Enterprise Records, which allows the organization to manage the life cycle of the content more effectively.

*Federated search* provides a simpler eDiscovery process by allowing a single search to span multiple repositories, using a combination of index values and document content or keywords. Documents that are relevant to a legal action can then be placed on hold.

The ability to *hold* documents is an important feature in a federated records management system. When you search for discoverable content, you can place all matching content on hold. You can insure that the content will not be deleted either by the regular disposition process or manually by a Records Manager until the hold is lifted (when the litigation event is terminated, or the content is no longer necessary).

### eDiscovery

eDiscovery helps to decrease the cost of electronic discovery, balance in-house implementation with outsourced components, and proactively manage electronically stored information. IBM offers the following three products as an alternative to federate and integrate records:

- *IBM eDiscovery Analyzer* provides early case assessment, conceptual analysis, and information insight capabilities for agile eDiscovery.
- *IBM eDiscovery Manager* helps you respond to litigation with agility by using a scalable, repeatable, in-house solution.
- *IBM Content Collector Discovery Analytics* provides complete collection, archiving, eDiscovery, and analytics capabilities in one solution.

Leveraging the IBM ECM platform, eDiscovery manages content within ongoing business processes and helps to increase defensibility in a security-rich, consistent, and auditable manner. It helps to reduce costs and risks while improving ongoing operational benefits.

For more information about eDiscovery, go to this Web site:

http://www.ibm.com/software/data/content-management/products/ediscovery

### 1.7.3 Records federation or collection

As a best practice, we recommend a careful analysis to determine whether to collect records from other repositories or to federate them. Not all source repositories offer capabilities to support effective and sound records management. Certain repositories do not scale. Several repositories do not provide the necessary controls to limit user behavior. Specific repositories are temporary in support of a specific project; however, the records from that project might have a long retention period.

Organizations must review the capabilities of source repositories and the associated products, particularly the following capabilities:

- Performance and scalability
- Preservation features
- Integration services
- Disaster recovery
- Operational capabilities
- Infrastructure support
- Security and access
- Situational factors

#### Performance and scalability

Review the performance and scalability capabilities:

- Performance: Does your repository require support for 20 million record transactions on a daily basis (for example, declarations, reviews, transfers, holds, destruction, and audits)?
- Scalability: Does your repository require the accommodation of billions of records and thousands of users across an enterprise?

#### Preservation features

Review the preservation features that are necessary:

- The product needs to be able to prevent an unauthorized delete.
- The product needs to preserve the content:
  - The product needs to insure that the content is preserved, trustworthy, and reliable.
  - The product needs to be able to minimize the risk of unauthorized changes to the content.

Archived

- The product needs to preserve the structure: The structure needs to remain (that is, the calculations and the format are preserved throughout the life of the record).
- The product needs to preserve the context:
  - The product must have the ability to insure the preservation of links (that is, e-mails with attachments and records with people, events, dates, and other attributes).
  - The product must have the ability to insure the preservation of the records and the processes that created and managed them.
  - The product must have the ability to preserve other records and logical groupings or mappings against objects in the file plan.
- The product must be able to participate in *hold* requests and related *hold* activities.

### Integration services

Review the integration services capabilities:

- The product has open, supported APIs to which a records management application can integrate with standard, ready to use integration (not custom services).
- The product's APIs support making a content item immutable.
- The product's APIs support the destruction of content.
- The product's APIs support the movement of content.
- The product's APIs support putting an item on hold, including additional elements of a hold management program that include create, apply (dynamically and through inheritance), notify, manage, review, report, and audit.
- The product's API must be versatile (so that metadata can be automatically collected and updated from authoritative systems).
- The product supports or is intended to support in the future standard access methods to access content objects, including industry standard efforts, such as common management information service (CMIS) and Electronic Discovery Reference Model (EDRM) XML.
- The product must have the ability to support multiple search engines and have access to search tools through open and industry-standard APIs.

### Disaster recovery

Review the disaster recovery capabilities:

- The product must support high availability.

Archived

- The product must support the Vital Records program.
- Scale-out: The product must be able to scale out without redefining a duplicate set of rules for any of these capabilities.

### Operational capabilities

Review the operational capabilities:

- The product supports process management for records processes.
- The product supports automated records declaration and classification.
- Searching: The product supports robust search features.
- Authorized users must be able to add and modify metadata (so that records transactions can be automated).
- Support for migration of content must exist.
- Support for rendition of content must exist.
- Support for electronic signatures and annotations must exist.
- Capabilities must exist to limit, audit, and monitor systems administrator support activities.
- Support for the de-duplication of content (for better storage management) must be available.

### Infrastructure support

Review the infrastructure support capabilities:

- Support of multiple kinds of storage must exist, including support from multiple vendors, as well as disk, tape, optical, and WORM.
- Support for hierarchical storage management (HSM): The movement and recall of information to and from various storage devices to facilitate better storage management must exist, in a way that is seamless to the user and application.
- If a relational database is used, support for multiple leading relational databases must be available.
- If Web server components are used, support for multiple Web servers must exist.
- Support for multiple operating platforms, including Windows®, UNIX®, Linux®, System i®, and System z®, must be available.

Archived

### Security and access

Review the security and access capabilities:

- The product needs the ability to control access to content to prevent unauthorized or unintentional disclosure.
- Auditing and monitoring: You need the ability to have full audit reporting and monitoring of all activities against the repository, with abilities to prevent tampering with or circumventing the repository.
- The product must have capabilities to limit, audit, and monitor systems administrator support activities.

### Situational factors

Review the situational factors:

- Make sure that the product roadmap is aligned with the organization's goals and objectives.
- Verify that the rules and the internal processes to manage systems are enforced uniformly throughout the organization.
- Make sure that the life of the host application exceeds the length of time that is required to preserve the content.
- Limit the number and scope of repositories—criticality of mass and content.
- Moving content is prohibited by local laws and regulations.
- Verify that the organization has a long-term strategy and a commitment for maintaining and supporting the product.

## 1.8 Federated business process management

A typical organization has hundreds of business processes at work. Managing these processes in the most efficient and effective manner is critical to the organization's success. Many of these processes involve people and documents. Users might need to access multiple systems and applications to get the information that they require to make decisions.

CFS helps to simplify the user experience by providing a single view into the content. Federated business process management adds the ability to activate the content from the source repositories and include it in business processes.

IBM FileNet Content Manager's event driven architecture allows events, such as adding a document to launch a process in IBM FileNet Business Process Manager, through event subscriptions. An *event subscription* is the association of a particular event trigger with an event action. For example, when you add a

document to a system, this created action is the event trigger, and it launches an associated business process.

Multiple subscriptions can be associated with one event trigger. For example, you can set up a system so that adding a policy document to Image Services does not trigger a workflow but adding a new claim document to Image Services can. The newly launched workflow can attach the claim document, as well as the customer's policy documents, to the workflow.

There are subtle distinctions among the three types of CFS and how the actions taken on content in the source repository can trigger workflows. For CFS-IS and CFS-CMOD, adding, deleting, or changing properties on a document within the source repository can trigger workflows. In the case of CFS implementation (using Content Integrator), adding or changing a document's properties in the source repository can trigger a workflow. Deleting a document in the source repository does not trigger a workflow.

In addition to being able to launch a workflow, federated content can be made available to existing processes. For example, a workflow might require the user to consult one or more documents before making a decision. Federated content can be attached to the process map, making it easy for the user to consult and make decisions.

Federated business process management plays an important role in records management. An organization can opt to declare records as part of a business process. For example, you can declare a document as a record when it completes a review and approval business process.

Alternatively, when a record is ready for disposition, you might want to send the documents through a review and approval process before their final deletion. In this scenario, Enterprise Records can automatically launch a disposition workflow and route the documents to a final review process before they are destroyed. The specific documents are attached to the workflow, and a reviewer can view these federated records (which might have source documents from separate repositories) easily to make a decision.

Archived

Archived

# Part 2

# IBM Content Federation Services

Archived

© Copyright IBM Corp. 2010. All rights reserved.

Archived

2

# Content Federation Services architecture

IBM Content Federation Services, also known as IBM FileNet P8 Content Federation Services, are available through IBM FileNet Content Manager (the product). In this chapter, we discuss Content Federation Services from a functional and conceptual standpoint. IBM offers three Content Federation Services implementations, and they share many common features and characteristics. This chapter focuses on those aspects that are common to all three implementations. For the remainder of this chapter, we refer to the three implementations collectively as CFS.

We cover the following topics in this chapter:

- Master catalog
- Content Federation Services architecture

© Copyright IBM Corp. 2010. All rights reserved.

## 2.1 Master catalog

Content Federation Services is based on the federation implementation strategy for managing distributed content. The distinguishing feature of this strategy is the master catalog. The master catalog is a searchable database that contains information about content that is stored in various repositories and repository types in separate locations throughout the enterprise.

**Clarification:** Do not construe the term *master catalog* to imply that external repository catalogs have a subordinate relationship to the CFS catalog. The master catalog is only the *master* in that it combines information originating from the other catalogs. It holds no special precedence otherwise.

### 2.1.1 Similarities between CFS and Web search engines

One way to understand the role of the master catalog in the CFS architecture is to consider how a Web search engine works. When a person using a Web browser navigates to a search engine site to perform a Web search, the person is not actually searching the Web. Instead, the person searches a centralized index that was constructed and is continuously updated by the search engine.

The index is created by a component of the search engine that gathers information from Web pages that are distributed throughout the Web and creates entries for these Web pages in an index. The index entries do not include the Web page itself, but instead, they consist of information about the Web page, that is, *metadata*. Index entries also include a reference to the location of the Web page so that the Web page can be retrieved.

Although there are many important differences, Content Federation Services is similar to a Web search engine in that they both provide searchable databases that contain information about content that is stored in a variety of remote repositories. (In the case of search engines, the remote repositories are all HTTP servers.) The advantage of this approach is that queries can be executed much faster and more efficiently, because they are executed against a single database rather than being distributed across multiple repositories. You do not need to aggregate the results, because there will always be only one result set.

Much like a Web search engine, CFS also includes functionality that is used to discover content in other repositories and to create catalog entries. Clients can then locate and retrieve content that is stored in these repositories by using the search and retrieval capabilities that are provided by the master catalog.

## 2.1.2 Using an object store as the master catalog

The CFS master catalog is implemented as a FileNet Content Manager object store. In terms of the object store schema, CFS master catalog entries are created as documents. We refer to the documents that are created in this way as *federated documents*.

The major difference between a federated document and a non-federated document is that a non-federated document has content that is directly managed by FileNet Content Manager as opposed to a federated document that contains a reference to the content in an external repository.

Figure 2-1 illustrates a federated document and its relationship to the source document in the repository from which it was federated.

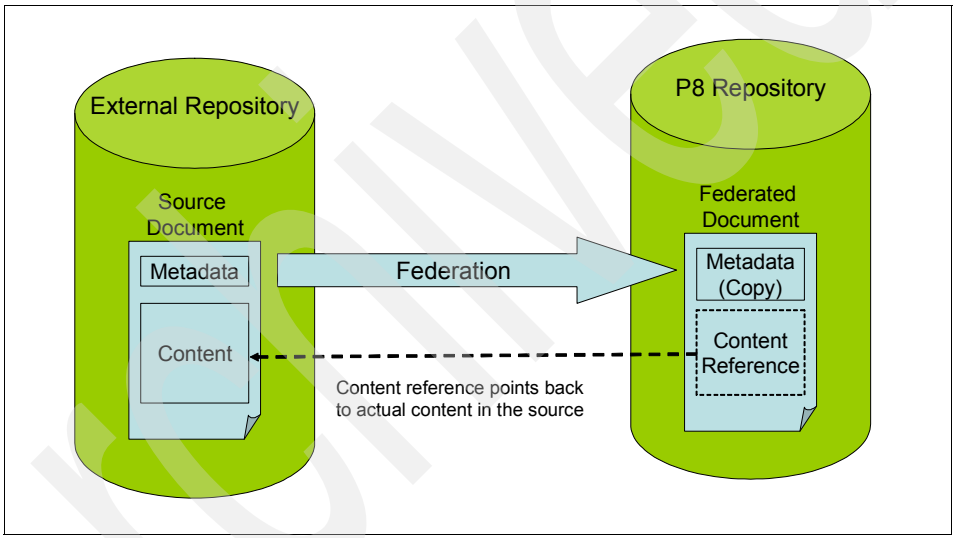



*Figure 2-1 Relationship between a federated document and a source document*

Among the benefits of using a FileNet Content Manager object store is that it allows a far richer set of metadata to be captured than is typically supported by Web search engines. And, as a result, content that is federated by CFS can be more fully described. The FileNet Content Manager object store schema is also extensible and allows customized behavior to be associated with it that can be used to enforce application-specific business rules.

Another important difference between Web search engines and CFS is that Web search engines return Web page references to the client in the form of URLs, and it is up to the client, which is typically a Web browser, to retrieve the page from the specified location. In the case CFS, the server is responsible for fetching the

Archived

content, thus relieving the client from needing to know where or how content is stored.

One of the great advantages of this approach is that federated documents are treated as first-class objects by FileNet Content Manager, meaning federated documents are no less functional than non-federated documents. The outcome is that the FileNet Content Manager API can function as a common API for working with content that is stored in any repository type that is supported by CFS. More importantly, it exposes federated content to the full range of functionality that is provided by FileNet Content Manager.

The capability to treat federated documents the same as non-federated documents is an enormous advantage, because it means that any application that is built on the FileNet Content Manager API operates no differently on federated documents than it does on non-federated documents. The end result is that CFS effectively extends the reach of P8 applications, such as IBM Enterprise Records (formerly known as IBM FileNet Records Manager) and IBM FileNet Business Process Manager, to non-P8 repositories.

Not only does CFS extend the reach of existing P8 applications to non-P8 repositories, it extends the reach without disrupting existing applications that use those repositories. Because IBM FileNet Content Manager applications require no special modification when working with federated documents, applications that are native to the external repository can also continue operation without any knowledge that federation is occurring.

## 2.2 Content Federation Services architecture

In this section, we provide a general description of the CFS architecture. Our objective is to define common concepts and to establish a common vocabulary to use through the more detailed discussion of each of the three specific implementations in subsequent chapters.

### 2.2.1 Important terms

Table 2-1 on page 37 defines important terms that we use to describe the CFS architecture.

*Table 2-1 Terminology pertaining to CFS*

| Term | Definition |
|---|---|
| Document | A document is a type of object that is defined by the FileNet Content Manager schema that combines content and metadata. Documents support behaviors, such as versioning, containability (the ability to file it in a folder), and record declaration. |
| Metadata | Metadata is data that describes the content that is associated with a document. Metadata is strongly typed in order to facilitate indexing and business rule enforcement. |
| Content | Content is a variable length sequence of bytes that are associated with a document. Although content data typically has its own internal structure, this information is relevant only to the applications that consume it and is ignored by FileNet Content Manager. As far as FileNet Content Manager is concerned, the byte sequence is undifferentiated and the only part that matters is the byte ordering. Examples of content include image files, word processing files, and video clips. |
| External repository | An external repository is a content repository that is external to the FileNet Content Manager system from which documents can be federated. |
| Federated document | A federated document is an instance of a document that has the address of an external document instead of content. |
| External document | An external document is a document or a document-like object[a] that is stored in an external repository. |
| Source document | A source document is an external document that has been federated by CFS. The federated document contains a reference to the source document. |
| CFS user | The CFS user is the identity of the security principal. The exporter and the Content Engine server use the CFS user to log in to an external repository to perform operations, such as retrieving content and metadata for federated documents. |
| Source repository | The source repository is an external repository that includes documents that have already been federated. |

Archived

| Term | Definition |
| --- | --- |
| Import request | An import request is an intermediate form of a federated document while it is in the process of being federated. Import requests are created by the exporter and placed on the import queue to be processed by the importer. Import requests include metadata from the source document being federated and the address where the document can be found in the source repository. |

a. A *document-like object* is any object that has the characteristics and behaviors of a P8 document, that is, an object that combines content and metadata and supports behaviors, such as versioning. Many repositories support these objects, but the objects are not necessarily called documents.

### 2.2.2 High-level architecture

Figure 2-2 on page 39 provides a high-level representation of the P8 Content Federation Services architecture. Note that this diagram is identical to Figure 1-1 on page 15, except that it has been relabeled to identify the P8 components that perform the depicted roles.

Archived

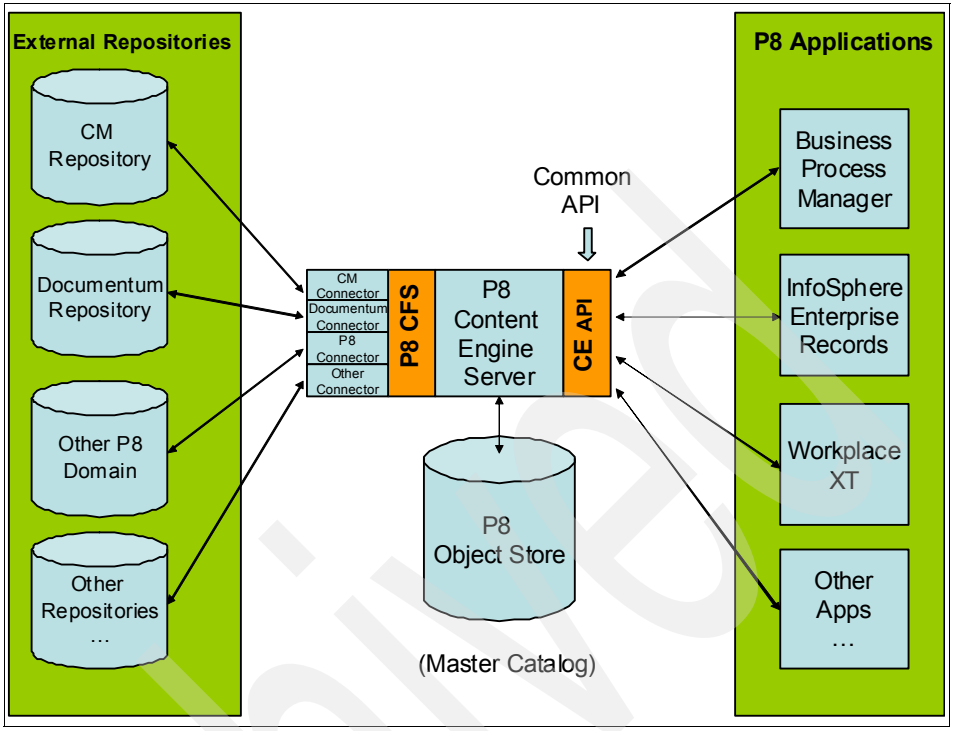



*Figure 2-2 Content Federation Services architecture*

Figure 2-2 shows a P8 object store that is performing the role of the master catalog. The Content Engine API implements the common API.

On the left side of the diagram, we see examples of the types of external repositories from which CFS can federate content. On the right side of the diagram, we see that all of the standard P8 applications can operate on federated content that is created by CFS.

### 2.2.3 Content Federation Services operations

CFS generally supports five operations with respect to federated documents: create, read, update, delete, and lockdown. We briefly describe each of these operations next.

### Create

A federated document is created when an entry is created in the P8 object store that has been designated as the master catalog. Master catalog entries are created as federated documents, which are similar to non-federated documents, except that a federated document has a reference to a document in an external repository instead of content.

Creation of a federated document occurs as part of *federation*, which is the process of discovering documents in external repositories and adding the documents to the master catalog. We describe this process in greater detail in the context of our discussion of CFS functional components in 2.2.4, “Content Federation Services functional components” on page 41.

### Read

Reading a federated document can mean retrieving the metadata or the content. In the case of a metadata read, reading a federated document is the same as reading a non-federated document, because the master catalog contains a copy of the metadata that is obtained from the external repository. However, when the content is read, P8 must use the reference that is stored as part of the federated document to locate the source document in the external repository and retrieve the document’s content. From a P8 client application standpoint, however, this redirection is completely transparent, and the application does not need to know that it is working with federated content.

### Update

Updates to federated documents are unidirectional; that is, updates that occur in the source repository are propagated to P8, but updates in P8 do not get propagated back to the source repository. Also, as with reads, there are two kinds of updates: metadata updates and content updates. *Metadata updates* occur in place so that when change is detected to the source document, the equivalent property on the federated document is overwritten with the updated value. *Content updates* require that a new version of the document is created. When a new version of a source document is created, an equivalent new version is created in P8.

In both cases, updates to P8 are performed by federation. When a source document that has been updated is refederated, CFS detects that the source document is an existing document. Then, CFS performs the necessary updates in P8, including creating new versions when necessary.

### Delete

When a federated document is deleted from P8, the delete is also propagated to the external repository to delete the source document. Also, delete is fully bidirectional for Content Federation Services for Image Services (CFS-IS) and for Content Federation Services for IBM Content Manager OnDemand (CFS-CMOD) but not CFS implementation (using IBM Content Integrator). Therefore, when the source document is deleted from either the Image Services source repository or the CMOD source repository, the delete is propagated to P8 where the federated document is also deleted.

### Lockdown

*Lockdown* is a specialized operation intended to allow federated documents to be declared as records by Enterprise Records.

When a document is declared a record, P8 must guarantee that it cannot be changed or deleted, except by the Records Administrator, Records Manager, or the records management disposition process. For federated documents, enforcing this guarantee requires cooperation from the external repository because that is where the content is actually stored.

Lockdown is the way that P8 notifies the external repository that the source document has been declared a record. When lockdown is called, the source repository must take actions to guarantee that non-authorized persons cannot update or delete the specified document using the external repository's native API.

Lockdown is never invoked directly by a P8 client and is only called due to record declaration by Enterprise Records.

## 2.2.4 Content Federation Services functional components

In this section, we examine the internal architecture of CFS by decomposing it into its functional components. We describe the specific manner in which each of these components is implemented for each of the three CFS offerings in detail later in the book. For now, we focus on identifying the functional components, describing what they do, and how they relate to each other in terms of data flow.

In this discussion, we emphasize the logical, rather than the physical, characteristics of the components. Therefore, the components that we identify do not necessarily map to the physical modules that we describe in subsequent chapters, which address each of the specific CFS implementations.

Archived

We discuss the CFS functional components in two parts. The first part describes the components that are involved with *federation,* that is, the process of discovering content and creating catalog entries in the master catalog. The second part describes the components that are related to content retrieval, deletion, and lockdown.

## Federation components

Figure 2-3 is an exploded view of the functional module labeled P8 CFS from the previous diagram, Figure 2-2 on page 39. It depicts the functional components of CFS that are associated with federation. A brief description of each component follows.

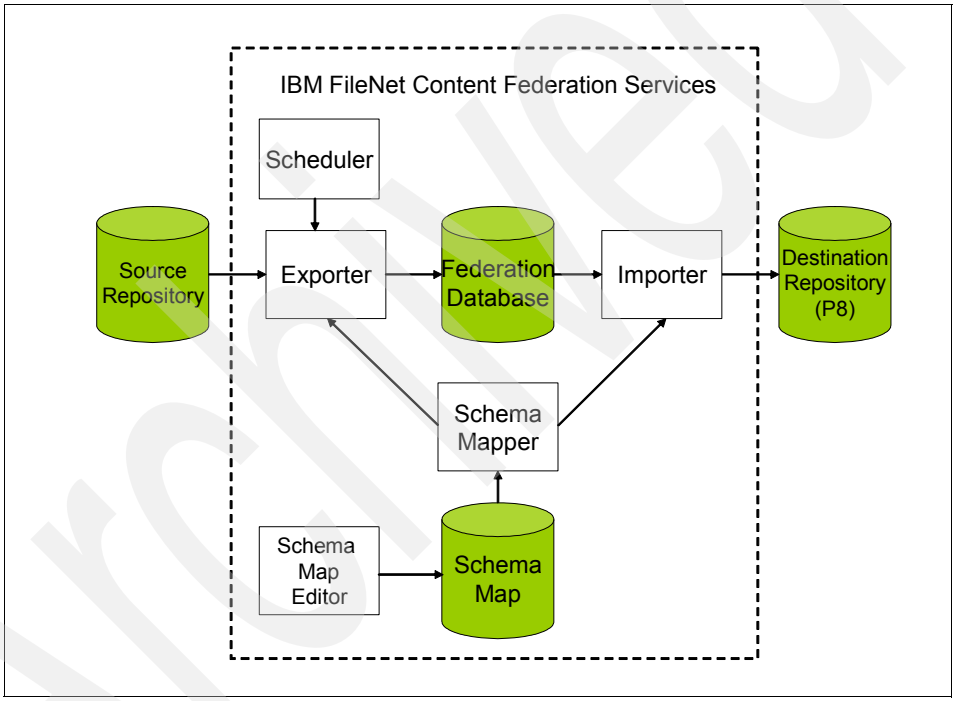



*Figure 2-3 Content Federation Services conceptual model*

### *Source repository*

The *source repository* is the external content repository from which source documents are federated. The source repository contains the actual content to which federated documents refer.

Archived

### *Destination repository*

The *destination repository* is an object store that is managed by FileNet Content Manager that is used as the master catalog by CFS. The destination repository is where federated documents are created.

Sometimes, the destination repository is also referred to as the *target repository*.

### *Exporter*

The *exporter* is the component of CFS that is responsible for discovering information about content in external repositories from which it creates import requests and places them on the import queue in the federation database.

### *Importer*

The *importer* is the component of CFS that reads import requests from the federation database import queue and creates or updates federated documents in the master catalog.

The import process insures that multiple import requests for the same document are processed in the correct order.

### *Federation database*

The *federation database* serves two purposes. It provides storage for the import queue, and it keeps track of all documents that have been federated. The latter is necessary to prevent duplicate federated documents that refer to the same source document.

The import queue provides an intermediate storage area for federation import requests that have been created by the exporter component. The import queue provides data persistence so that the importer and the exporter can execute independently while guaranteeing the durability of import requests in the case of system failure.

### *Schema mapper*

The *schema mapper* is the component of CFS that is responsible for transforming source document types and metadata into the equivalent FileNet Content Engine classes and properties as defined by the schema map.

Depending on the implementation, either the exporter component or the importer component can invoke the schema mapper.

### *Schema map*

The *schema map* relates the schema objects that are defined in the source repository, such as document types and properties, to the semantically equivalent schema objects in the FileNet Content Manager object model.

Archived

Document types in the source repository are always mapped to instances of the document class or to one of its subclasses in the destination P8 repository.

For example, consider a source repository that represents document objects as *items* and metadata as *attributes*. Items have content and attributes, and the attributes describe that content. In this example, an item is equivalent to an instance of a P8 document, and an attribute is equivalent to a property.

Now, suppose that the source repository in our example defines a particular type of item that is called a Purchase Order, and the Purchase Order item type has attributes that include a Purchase-Order-Number and a Customer-Id. In order to federate Purchase Order items from the source repository to P8, a document class must be defined in the destination repository that is the semantic equivalent to the Purchase Order item type. The Purchase Order document class must have properties that are equivalent to the Order Number and Customer Id attributes.

It is not enough, though, to have equivalent schema objects that are defined in the source and destination repositories; there must be a way to tell CFS how they are related, which is the purpose of the schema map.

In this example, the schema map includes a mapping of a Purchase Order item type to the Purchase Order document class. The mapping also includes information that relates the Purchase-Order-Number and Customer-Id attributes to the equivalent properties on the Purchase Order.

The names of the source schema objects and their equivalent P8 schema objects can be the same, as in our example. However, this approach is unnecessary, and it is often impossible. The only requirement is that the data types must be compatible.

### Schema map editor

The *schema map editor* is the component that provides the user interface to allow administrators to create and edit the schema map.

### Scheduler

The *scheduler* is the component that is responsible for initiating the exporter based on a specific triggering event. The scheduler can be poll-based, in which case, the triggering event is simply the expiration of a timer. It can also be more purely event-based; in which case, the triggering event is a transaction that occurs in the source repository, such as the creation of a document of a certain type.

## Federation process

*Federation* is the process of discovering content in an external repository and creating entries for it in the master catalog. The federation process consists of these steps:

1. The scheduler detects a triggering event, such as the expiration of a timer, and starts the exporter.
2. The exporter searches the source repository to discover external documents to federate. The method that the explorer uses is implementation-dependent. Discovery can be based on a query that specifies complex selection criteria, or it might be as simple as looking in a well-known location and federating all of the documents that it finds there.
3. The exporter copies the metadata that is obtained from the external documents that it finds. It marshals the metadata into one or more import requests that are placed on the import queue in the federation database. In addition to the metadata, the import request also includes the address of the source document. The exporter might invoke the schema mapper at this time, depending on the implementation. If it does not invoke the schema mapper, the responsibility for invoking the schema mapper belongs to the importer.
4. The importer processes requests from the import queue. The importer creates catalog entries for each newly federated document or document version that it finds in an import request. If the federation request specifies a document that already exists in the destination repository, the existing document is retrieved and its properties are updated. If the federation request specifies a new version of the external document, a new version in the corresponding P8 version series is created.

   Creating a catalog entry for a newly federated document consists of creating an instance of the P8 document class that is equivalent to the source document type as defined by the schema map. After the new document instance is created, its property values are populated from the import request.

   Creating a catalog entry for a new version of a federated document consists of performing a checkout of the previous federated document version. Property values from the new external document version are copied. Then, the federated document is checked back in, thus, creating a new version in P8 that corresponds to the new external document version.

   The document and document version instances are created in the P8 object store as federated documents, which means that instead of having content, they have a reference to the source document or source document version. The reference to the source is synthesized from data in the import request.

Archived

## Content retrieval components

Figure 2-4 depicts the functional components of CFS that are associated with content retrieval. Then, we briefly describe each component.

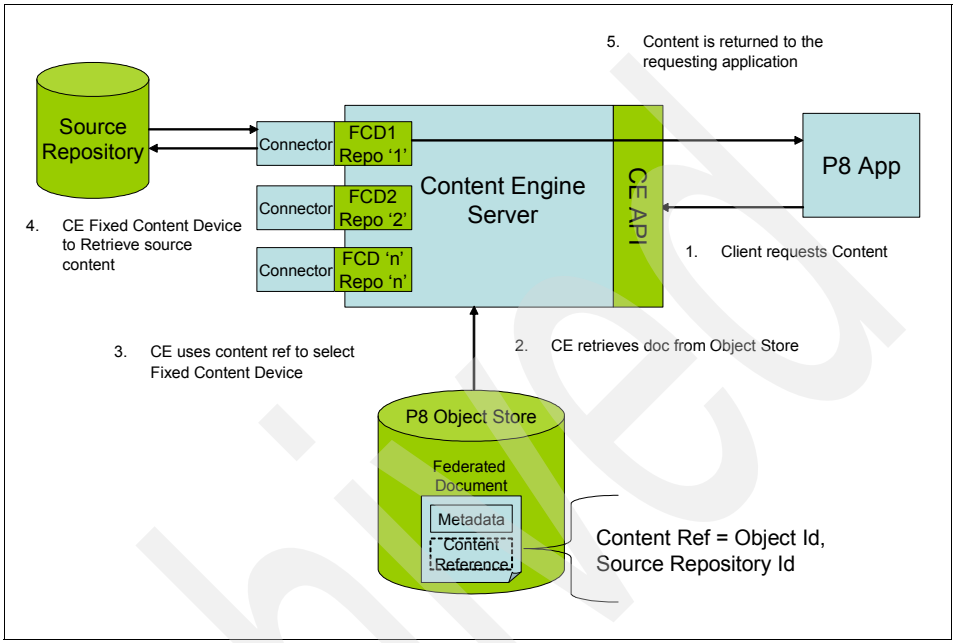



*Figure 2-4 CFS content retrieval components*

### *Fixed content device*

The *fixed content device* is a P8 object that represents an external repository. It has metadata that specifies connection information to the repository, including the credentials for the CFS user. Fixed content devices have connectors that are associated with them that are used to provide the actual interface with the external repository.

Fixed content devices are scoped to the P8 domain, and a single domain can have multiple fixed content devices defined.

> **Note:** The fixed content device derives its name from the fact that its original purpose was to represent read-only and write-once content storage devices. They were later adapted for use by CFS; however, because fixed content devices are still used for their earlier purpose, as well as by CFS, they retained their original name.

Archived

### *Fixed storage area (not shown)*

All FileNet Content Manager documents have a storage area that is associated with them that identifies the location of the content. A *fixed storage area* is a type of storage area that indicates that the storage area is actually an external repository. Fixed storage areas are associated with an instance of a fixed content device that represents the specific external repository where the content is actually stored.

### *Content Engine API*

The *Content Engine API* is the interface that the FileNet Content Manager exposes to application programs from which the FileNet Content Manager can perform operations on documents and other objects that are defined by the repository schema. The Content Engine API makes no distinction between federated documents and non-federated documents and, thus, eliminates the need for applications to be aware of whether they are working with federated documents.

### *External content reference*

An *external content reference* is the address of an external document that has been federated. The external content reference includes both the identity of the repository and the identity of the source document within that repository. External content references appear instead of the actual content on federated documents.

### *Federated document*

A *federated document* is an instance of a document that has the address of an external document instead of the content.

### *Connector*

A *connector* is a layer of software that functions as an interface adapter between a specific external repository's native API and the higher layers of CFS. Connectors are used to normalize access to various types of external repositories. The connector encapsulates all repository-specific dependencies for a particular repository type. A separate connector implementation exists for each repository type.

## Content retrieval process

The process of retrieving federated content from the source repository consists of these steps (refer to Figure 2-4 on page 46):

1. An application that is based on the FileNet Content Manager API makes a request to retrieve the content for a specific document. Nothing distinguishes the federated content from the non-federated content. So, from the application's perspective, the application does not need to be aware that it is requesting content that is actually stored in an external repository.

2. The Content Engine server retrieves the document that is associated with the request and detects that the content is federated based on the document's storage area, which lets the server know that the content is really a reference and not the actual content. The Content Engine server retrieves the reference and decodes it to determine the identity of the source repository.
3. The Content Engine server uses the source repository identity to select the appropriate fixed content device to use to access the content. It then opens the document by passing the address of the document to the fixed content device.
4. The fixed content device logs into the external repository as the CFS user using its associated connector. The fixed content device then fetches the content for the specified document.
5. The content is returned to the requesting application as a stream of bytes.

### Content deletion process

The process of deleting federated content from the source repository uses the same components as the content retrieval process and follows these steps:

1. An application that is based on the FileNet Content Manager API makes a request to delete a specific document. Nothing distinguishes the federated documents from the non-federated documents. So, from the application's perspective, it does not need to be aware that it is deleting a document whose content is actually stored in an external repository.

   The Content Engine server accesses the document that is associated with the request and detects that the content is federated based on the document's storage area. The Content Engine server then verifies that the document can be deleted. If it passes all of the deletion criteria, the document is deleted from the P8 object store, and a deletion request for the document in the external repository is added to a persistent queue in the Content Engine server.

   After the document has been deleted from the P8 object store, the P8 API client's transaction is considered committed. So, this delete in the external repository is guaranteed to occur. Using a persistent queue in the Content Engine is the implementation to retry transient delete failures in the external repository.
2. The Content Engine server processes these deletion requests in a background task. When it picks up a deletion request to process, it determines the identity of the source repository.
3. The Content Engine server uses the source repository identity to select the appropriate fixed content device to use to delete the content.

Archived

4. The fixed content device logs in to the external repository as the CFS user using its associated connector. The fixed content device then deletes the content for the specified document.
5. If the delete operation fails due to a transient condition, such as a network failure, the delete request is put back in the Content Engine persistent queue to be retried later.
6. Delete requests are retried periodically until the request succeeds. After the delete request succeeds, it is removed from the persistent queue.

### Content lockdown process

The process of locking down federated content in the source repository uses the same components as the content retrieval process and consists of these steps:

1. Enterprise Records makes a request to lock down a specific document.
2. The Content Engine server accesses the document that is associated with the request. Depending on the CFS implementation, the lockdown operation is processed either synchronously, such as content retrieval, or asynchronously, such as document deletes.
3. For CFS implementation (using Content Integrator) and CFS-CMOD, lockdown operations are processed synchronously:
   a. The Content Engine server determines the identity of the source repository and selects the appropriate fixed content device to lock down the content in the external repository.
   b. The fixed content device logs in to the external repository as the CFS user using its associated connector. It then locks down the content for the specified document.
   c. If the lockdown operation succeeds, a success is returned to Enterprise Records.
   d. If the lockdown operation fails for any reason, even for transient network reasons, a failure is returned to Enterprise Records.
4. For CFS-IS, lockdown operations are processed asynchronously:
   a. Lockdown requests are added to a persistent queue in the Content Engine server to guarantee that they will be executed. Adding a lockdown request commits the transaction, so a success is returned to Enterprise Records.
   b. The Content Engine server processes these lockdown requests in a background task. When the Content Engine server picks up a lockdown request to process, it determines the identity of the source repository.
   c. The Content Engine server uses the source repository identity to select the appropriate fixed content device to use to lock down the content.

Archived

d. The fixed content device logs in to the external repository as the CFS user using its associated connector. It then locks down the content for the specified document.

e. If the lockdown operation fails due to a transient condition, such as a network failure, the lockdown request is put back in the Content Engine persistent queue to be retried later.

f. Lockdown requests are retried periodically until the request succeeds. After the lockdown request succeeds, it is removed from the persistent queue.

### 2.2.5 Security of federated documents

When documents are federated, they inherit the security of the P8 document class to which they have been mapped. Thus, when performing the schema mapping task, insure that the security of the P8 document class is equivalent to the security of the corresponding object in the external repository.

Users of P8 applications access federated documents using the security that is provided in P8. This security might allow separate access rights for a user, depending on whether the user accesses the document using a P8 application or a native application on the external repository.

For example, when Enterprise Records declares a document a record, access to the content from P8 is controlled by the file plan location where the record is filed. However, this security does not cascade to the source repository where the content is locked down and prevented from being updated or deleted. Thus, the set of users that can view the document from P8 might differ from the set of users that can view the document in the source repository using its native applications.

In the following chapters, we describe in detail the three implementations of Content Federation Services, their architectures, their specific federation processes, and their implementations:

- Chapter 3, “Content Federation Services for Image Services” on page 51
- Chapter 4, “Content Federation Services for Content Manager OnDemand” on page 145
- Chapter 5, “Content Federation Services implementation using Content Integrator” on page 189

Archived

3

# Content Federation Services for Image Services

IBM FileNet P8 Content Federation Services for Image Services (CFS-IS) provides the ability to federate documents in multiple ways. Each way provides its own benefits and comes with challenges. It is important to plan and prepare before implementation begins so that you federate documents properly. In this chapter, we discuss how to implement CFS-IS most effectively in your Enterprise Content Management (ECM) environment.

We cover the following topics in this chapter:

- Architecture
- Federation process
- Planning
- Installation
- Configuration
- Federated records management with Content Federation Services for Image Services
- Troubleshooting
- Tuning
- Best practices

© Copyright IBM Corp. 2010. All rights reserved.

# 3.1 Architecture

The CFS-IS architecture that is described in this section expands on the CFS conceptual model that is discussed in Chapter 2, “Content Federation Services architecture” on page 33. We discuss the specific manner in which the CFS-IS offering implements the CFS conceptual model.

## 3.1.1 Mapping the Image Services data model to P8

Before discussing the architectural components of the CFS-IS implementation, we briefly describe how the Image Services (IS) data model maps to the P8 data model.

The data model that is used by Image Services is similar to the data model that is used by P8:

- An Image Services *document class* corresponds to a P8 *document class*.
- An Image Services *document* corresponds to a P8 *document*.
- The *indexes* of an Image Services document correspond to the *properties* of a P8 document.
- A *page* of an Image Services document corresponds to a *content element* of a P8 document.
- A P8 object store can contain multiple *document classes.* Hence, it can contain mappings to multiple Image Services *document classes* for that object store.
- An Image Services *document class* cannot be mapped to more than one P8 object store.

## 3.1.2 Components

Figure 3-1 on page 53 shows the CFS-IS components that implement the CFS conceptual model.

Archived

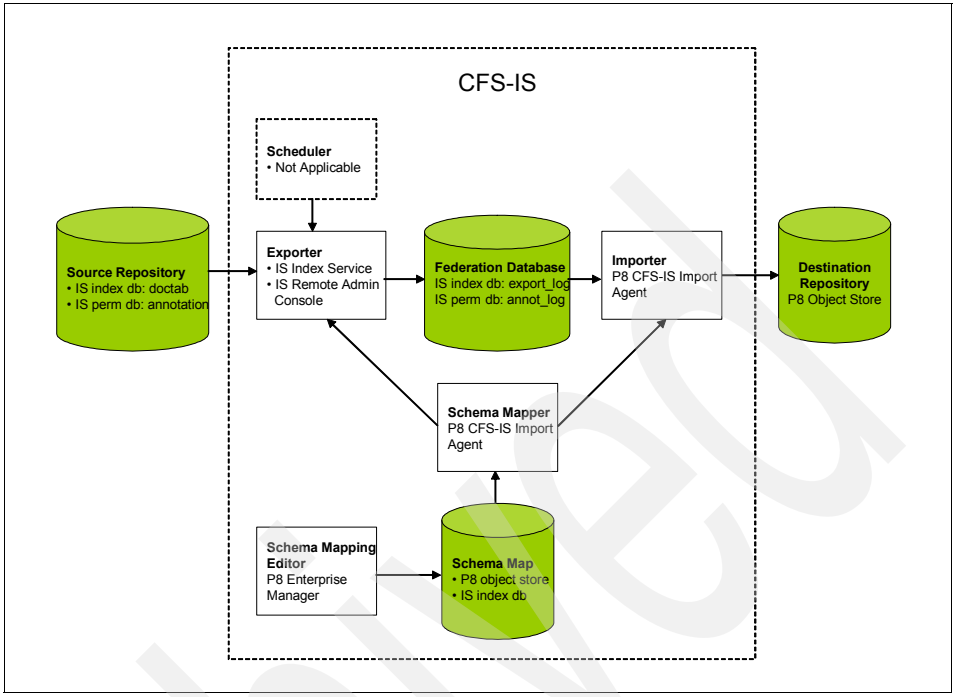



*Figure 3-1 CFS-IS components that implement the CFS conceptual model*

CFS-IS consists of these components:

- Schema mapping editor
- Schema map
- Scheduler
- Source repository
- Exporters
- Importer
- Schema mapper
- Federation databases
- Destination repository

In this section, we provide an introduction to these components. We discuss their use in 3.2, “Federation process” on page 55.

## Schema mapping editor

To prepare Image Services documents for federation, you create a *schema map* to map Image Services document classes and properties to P8 document classes and properties. This mapping occurs in IBM FileNet Content Manager (P8 CM) using the IBM FileNet Enterprise Manager.

### Schema map

The results of the schema mapping process are persisted in both the P8 object store and the Image Services server database.

### Scheduler

There is no scheduling involved with CFS-IS federation. Federation happens automatically as new documents are created by Image Services native applications. Image Services document classes can be assigned a default P8 object store to allow this automatic federation to occur.

### Source repository

The source repository for Image Services is the Image Services database. In particular, the following Image Services database tables contain the content for the federated documents:

- The doctaba database table (in the Image Services index database) is the document source repository.
- The annotation database table (in the Image Services Perm database) is the annotation source repository.

### Exporters

The exporters for CFS-IS are the Image Services Index Service and the Image Services Remote Administration Console (RAC). The Image Services Index Service provides the real-time exporting function as documents are created, updated, or deleted in Image Services. RAC provides a batch mechanism to manually export or re-export documents to a P8 object store.

### Importer

The importer for CFS-IS is the CFS-IS Import Agent that runs as part of P8 CM.

### Schema mapper

The schema mapper for CFS-IS is also the CFS-IS Import Agent that runs as part of P8 CM. The CFS-IS Import Agent performs the schema mapping function as documents are federated into P8 using the schema map that was created by the schema mapping editor, which we described earlier.

Archived

### Federation databases

The federation databases for Image Services are the following database tables on the Image Services server:

- The export_log table (in the Image Services index database) is the document export queue.
- The annot_log table (in the Image Services Perm database) is the annotation export queue.

### Destination repository

In all CFS implementations, the destination repository is the P8 object store.

## 3.2 Federation process

The CFS-IS federation process involves two distinct phases that are performed on the Image Services server and the P8 CM server: exporting and importing.

### Exporting

The exporting phase of CFS-IS occurs on the Image Services server. The Image Services export_log and annot_log database tables serve as an import queue for CFS-IS. The exporting phase of CFS-IS involves adding entries to the two tables that correspond to the Image Services documents and annotations that are being federated.

While Image Services documents are created, updated, or deleted, the Image Services Index Service uses the document class to determine if this document event must be federated. If so, the Image Services Index Service adds an entry to the export_log database table containing the metadata identified by the schema map associated with the document class.

Annotations that are added, updated, or deleted from any Image Services document that has been federated are federated, as well. These annotation events are added to the annot_log database table.

You can use the Image Services Catalog Export Tool within RAC to manually initiate the export of existing Image Services documents. This action calls the INX_export process in the background. INX_export then retrieves rows in the doctaba table that satisfy the selection criteria and put them in the export_log table. Annotations that are associated with these documents are entered in the annot_log database table.

Archived

### Importing

The importing phase of CFS-IS occurs within P8. The Image Services Import Agent, if enabled, runs continuously in the P8 CM server. The Image Services Import Agent periodically queries the Image Services server, which examines the export_log and annot_log tables for new entries. The Import Agent processes batches of rows at a time from these tables. After each batch has processed, it is removed from these tables.

The Image Services Import Agent performs the schema mapping function to map the Image Services document classes and properties to the P8 object model. Thus, it processes document create, update, and delete requests from the import queue and performs the corresponding action on the associated P8 document. For newly federated documents, it also creates the reference that is used by the P8 CM server to access the content for the document that is located on the Image Services server.

## 3.2.1 Federated document operations

To manage federated Image Services documents, you can perform several operations on these documents, including create, retrieve, lockdown, and delete.

### Create

The create operation is performed by the CFS-IS Import Agent. The following list describes the details of this operation:

- IS documents federated to P8 cause a corresponding P8 document to be created with the mapped properties of the Image Services document.
- The P8 document contains a reference to the content of the Image Services document that is located on the Image Services server.
- IS document pages correspond to the associated P8 document's content elements.
- Annotations of the Image Services document cause corresponding P8 annotations to be created.
- These P8 annotations are associated to the P8 document that corresponds to the federated Image Services document.
- Unlike federated document content, the content of the Image Services annotation is copied into the corresponding P8 annotation. There is no reference back to the annotation content on the Image Services server.
- The P8 annotation content is stored in the storage area that is selected by the storage policy property of the P8 ISAnnotation document class. By default, this storage area is the P8 object store database.

## Retrieve

You can retrieve the content of any federated Image Services document using P8 Workplace, WorkplaceXT, Enterprise Manager, or any custom user interface that is written to the P8 API.

When P8 retrieves federated content, it goes back to the Image Services server to get the native bytes. When P8 retrieves metadata, it returns from the P8 object store the mapped properties that were created or updated during federation.

## Update

The content of federated documents cannot change. Neither P8 nor Image Services allows the content of a document version to change after the document version has been checked in. Only the mapped properties can be updated. When property updates are made on the Image Services server for federated documents, the following conditions apply:

- Property updates of Image Services federated documents are propagated to P8, resulting in corresponding updates to the federated document's properties.
- If the Image Services document is re-exported, it overwrites all of the corresponding P8 document's properties, including the properties that might have been updated by a P8 application.
- Updates to Image Services annotations federated to P8 cause the previous P8 annotation to be deleted and cause a new P8 annotation with the updated content to be created.

When property updates are made by P8 applications for federated documents, the following conditions apply:

- If the P8 properties of the federated document are updated by a P8 application, it causes future updates of those properties from Image Services to fail. To prevent P8 applications from updating the P8 document properties of a federated Image Services document, adopt procedures that avoid updates on the P8 side or use P8 security features to prevent these updates.
- The P8 property updates are only visible to P8 applications. These updates are not propagated back to Image Services.
- Because federated annotation content is copied into P8, updates to these annotations by P8 applications are the same as updates to regular P8 annotations. These updates are only visible to P8 applications and are not propagated back to Image Services.

Archived

### Lockdown

To prevent updates by Image Services native applications to Image Services documents that have been federated, P8 applications, such as IBM Enterprise Records (formerly known as IBM FileNet Records Manager), can initiate a lockdown of the document on the Image Services server. The following list describes the details of this operation:

- Lockdown changes the security of the document (metadata and content) and associated annotations on the Image Services server to prevent updates or deletes to that document by native Image Services applications for non-administrators.
- Lockdown requires additional configuration changes to be made to the Image Services server. These changes involve creating a security group for lockdown, adding the CFS user as a member of that group, and specifying the security behavior when the lockdown operation is invoked by P8 using this CFS user.

### Delete

The delete operation of CFS-IS is bidirectional. The following list describes the details of this operation:

- Deletes of Image Services documents by Image Services native applications are propagated to P8 where the corresponding P8 document is deleted.
- Deletes of Image Services annotations by Image Services native applications are propagated to P8 where the corresponding P8 annotation is deleted.
- Deletes from P8 applications of federated Image Services documents are propagated to Image Services where the corresponding Image Services document is deleted.
- Deletes from P8 applications of federated Image Services annotations are propagated to Image Services where the corresponding Image Services annotation is deleted.

## 3.3 Planning

Before implementing CFS-IS, consider the following topics and how they relate to your environment. It is important to configure CFS-IS according to your specific needs.

Archived

### 3.3.1 Using Content Federation Services for Image Services

We list various use cases for document federation using CFS-IS. In certain cases, you can use a combination of the scenarios. The documents are created using either P8 or Image Services applications. The documents that are created with P8 applications are always native P8 documents. The documents that are created with Image Services applications but that are available to P8 applications are the federated documents.

#### New documents

The process of federating new documents differs somewhat from the process of federating existing documents. The following use cases are for federating new documents:

- Native P8 documents are added with FileNet P8 applications, such as Workplace, WorkplaceXT, and IBM Content Collector for eMail. The metadata is only stored in the P8 Content Engine, and the content is stored in an Image Services storage library. This use case is not considered document federation. Rather, it uses Image Services as a fixed content device to store content from P8. This use case is out of the scope for this book.
- Documents are added with Image Services applications, such as HPII, Capture, and Integrated Desktop Management and, then, federated to P8. The metadata is only stored in the P8 Content Engine, and the content is stored in an Image Services storage library. During committal, the Image Services metadata is not cataloged in the Image Services database. Documents can only be retrieved using P8 applications. You cannot retrieve the documents with Image Services applications.
- Documents are added with Image Services applications and, then, federated to P8. The metadata is stored in both P8 and Image Services and the content is stored in an Image Services storage library. Documents can be retrieved using P8 applications, such as Workplace, as well as with native Image Services applications.

#### Existing Image Services documents

The following use cases are for federating existing, or current, Image Services documents:

- During federation, the metadata of existing Image Services documents is copied to P8 and then removed from the Image Services catalog. The content remains in the Image Services storage library. Documents can only be retrieved by using P8 applications, such as Workplace, WorkplaceXT, and IBM Content Collector for eMail. You cannot retrieve documents with Image Services applications, such as Integrated Desktop Management.

- During federation, the metadata of Image Services documents is copied to P8. Both the metadata and the content remain in Image Services. Documents can be retrieved using P8 applications, as well as with native Image Services applications.

### 3.3.2 Your current Image Services system

The document life cycle and volumes on your Image Services system can impact the federation to P8. Consider the following aspects of your Image Services system:

- Number of documents in your Image Services repository
- Storage locations
- Document input methods
- Document retrieval methods
- Post-committal metadata updates
- Annotation use
- Document expiration process
- Document deletion
- Document security
- COLD documents
- Records management
- Remote locations

#### Number of documents in your Image Services repository

IS and Content Engine are both robust ECM systems, and CFS-IS can process large federation volumes, but it is best to only federate documents that are necessary. Image Services is primarily a search and retrieval system, and its database needs are less significant than P8's database needs. The Content Engine database contains much more information for each document than Image Services, so consider carefully before federating all documents from a large Image Services system. It can take a considerable amount of time to federate all Image Services documents to Content Engine, which can then result in an extremely large Content Engine database. Federate only the documents that you need.

#### Storage locations

Existing documents that are stored in Optical Storage and Retrieval (OSAR) and Magnetic Storage and Retrieval (MSAR) repositories can be federated using either of the two cases where documents are added with Image Services applications. Documents that are stored using the older Near Line Storage (NLS)-based protected storage models, such as CSAR (Centera) and SSAR (NetApp® Snaplock), require Image Services cataloging for their storage references. Although they can be federated with the use case where metadata

remains in Image Services after federation, they cannot be federated in the use case where the metadata is not cataloged in Image Services.

With Image Services 4.1.1 and later, you can store documents on protected storage systems using Integral Single Document Storage (ISDS), which does not require Image Services cataloging, thereby, eliminating the Image Services cataloging requirement on new protected storage documents. However, existing documents added with NLS must either be converted to ISDS format or their catalog entry must remain in the index database. In addition, using NLS is still possible with Image Services 4.1.1 and still requires the use of Image Services cataloging, but it is not practical when documents can be added with ISDS.

### Document input methods

When planning for future document input, consider that certain input methods will commit to Image Services but not to P8 and certain input methods will commit to P8 but not to Image Services. So, if you plan to continue adding documents with Image Services methods, there are few changes with federation. But, if you are planning to add documents with P8, your input methods must change.

Another important consideration if you plan to continue adding documents through Image Services is that any change in federated document classes or user-defined indexes must be replicated in P8. Otherwise, federation errors occur. For example, if you add a new menu item to an index in a federated Image Services document class, but you do not add a corresponding item to the mapped choice list in P8, the import process will fail when it tries to federate the document that uses the new menu item. So, with federation, an additional level of change control exists when making changes to document input methods.

### Document retrieval methods

The applications that are available to retrieve federated documents depend upon the use cases that were previously discussed. Be prepared to fully convert from Image Services applications (such as Integrated Desktop Management) to P8 applications (such as WorkplaceXT) if federated documents no longer have metadata stored in the Image Services catalogs.

### Post-committal metadata updates

If you make metadata changes to federated Image Services documents after the initial document committal, the changes will be propagated to P8 but only if those changes were made using Image Services tools (such as Integrated Desktop Management and Image Services Toolkit). Direct database changes using SQL commands or tools will not be detected by the Image Services daemon and will not, therefore, be propagated to P8. If this type of a change was made, the affected document must be re-exported by the Image Services export tool.

Archived

If metadata remains in Image Services, especially if you plan to make metadata updates in the future, all updates must be made using Image Services tools. If you make changes in P8 to the metadata of federated documents, the catalogs will become out of sync, and the future federation of the affected documents will cause CFS-IS to fail. To resolve this situation, you can manually update the changes in Image Services or re-export the documents in Image Services. Remember re-exporting will overwrite the P8 metadata with the Image Services metadata. For these dual-catalog systems, you can avoid issues by implementing procedures that restrict updates in P8 or by using P8 security to prevent these updates, such as creating marking sets, which are described later in this chapter.

### Annotation use

If you want to federate existing and future annotations, you must perform several additional steps, which we describe in later sections.

### Document expiration process

By default, Image Services will not export closed documents. If you want to federate closed documents, you must perform several additional steps. We describe these steps in later sections.

### Document deletion

Bidirectional document deletion is possible with CFS-IS. While bidirectional deletion is a feature, it is also a risk if not considered carefully. See more about bidirectional deletion in 3.3, “Planning” on page 58.

### Document security

CFS-IS uses a single user ID to access documents in Image Services. This ID is specified during the Content Engine CFS-IS configuration. This ID must be a member of the Image Services SysAdmin group or another group with read, write, and append/execute permissions to all of the mapped Image Services document classes.

### COLD documents

If you plan to federate COLD documents, be sure to include the associated COLD templates so that the P8 viewer can display the documents correctly.

Do not configure the Image Services export process to delete the catalog entry for the COLD templates, because this configuration causes problems for future COLD document committals.

Archived

### Records management

You can lock down federated Image Services documents with Enterprise Records (formerly known as IBM FileNet Records Manager), but additional steps are required. We describe these steps later.

Marking sets can be useful in preventing property updates on P8 documents, but they must not be used with Enterprise Records. Enterprise Records requires the ability to update metadata to declare a document as a record.

### Remote locations

If the Image Services and the Content Engine are in separate locations, especially if they are over a wide area network (WAN), you can locate an Image Services cache on the same server or local area network (LAN) as the Content Engine. This approach can dramatically improve the P8 retrieval performance of federated documents. We describe this concept in more detail later in this chapter.

## 3.3.3 Image Services objects to include in federation

When you are planning your federation, consider how much of your current Image Services system you want to federate. The following list of objects is available to federate to P8. Determine the extent that you want to federate each object:

- Domains
- Document classes, including volumes of each document class
- User-defined indexes
- System indexes
- Annotations and annotation security

## 3.3.4 Planning considerations

The following areas can help you to configure your system appropriately:

- Metadata updates
- Locating federated documents
- Document unfederation
- Metadata mapping: Content Engine properties and Image Services indexes
- Data considerations
- Other mapping considerations
- Content-based retrieval
- Closed Image Services documents
- Versioning
- Business process management

Archived

- Post federation migration

## Metadata updates

You must not, as part of normal business processes, update properties in both Image Services and P8 for a single federated document class.

If you need to change properties on a dual-catalog federation, you must almost always make the change in Image Services. The updates will be propagated to P8. However, if you are sure that you will never need to make metadata changes in Image Services, you can make the changes in P8.

If updates are made in P8, further updates from Image Services to P8 for that document fail. You can resolve this issue by manually updating the property or by re-exporting the document using the Image Services export tool.

During a transition, you can implement temporary procedures that restrict updates in P8 for CFS-IS documents or use P8 security to prevent these updates, such as creating marking sets. The steps to create marking sets are described later in this chapter. However, this type of marking sets is not possible if you use Enterprise Records, because Enterprise Records requires the access to change certain properties.

## Locating federated documents

When you federate Image Services documents to P8, the documents are not filed in a folder, so you cannot find federated documents within Workplace or WorkplaceXT by browsing a folder. However, within FileNet Enterprise Manager, the documents are visible in the unfiled documents folder within the object store. The unfiled documents folder is not an actual folder but a virtual folder. These documents are logically unfiled, which means that there is no link to a folder object in the database for that document. When a document is filed in an actual folder, it has a link to a folder object. Unless the volume of federated documents is extremely small, do not attempt to view documents in the unfiled documents virtual folder, because it might take a long time to display all of the documents.

You can use an event subscription to automatically file newly federated documents into a folder, but you might not have a real need for folders. An alternate method to find federated documents is by searching using a query tool, such as WorkplaceXT search applets.

You can also search using FileNet Enterprise Manager. For all searches, it is important to utilize search criteria to focus the results on a small number of documents. You can search more efficiently by ensuring that a database index is available that allows the database server to quickly locate the desired rows in the destination repository.

As part of your federation planning, you must insure that Image Services indexes that might be used to locate documents are properly mapped to Content Engine properties in the associated document classes and are made available to users for searching.

## Document unfederation

After documents have been federated, you cannot unfederate them using P8 applications. You cannot delete a federated document from either Image Services or Content Engine without causing a corresponding deletion on the other system. Often, without understanding this concept, clients set up a test of CFS-IS functionality on a test P8 system and federate a sample set of documents from a production Image Services system. Then, when the test is over and the functionality is proven, they attempt to remove the federated documents only from Content Engine and realize that they cannot, at least not with normal P8 tools.

If you want to test this situation, you can create two object stores: one object store for testing and one object store for production. Then, create one map from the Image Services production documents to the test object store and another identical map to the production object store. You can federate production documents into the test object store, and after confirming that the federation is successful, you can federate (re-export) the same production documents into the production object store. This step breaks the link between the test object store and the Image Services documents. You can now safely delete the documents from the test object store. When this delete request comes into Image Services, it is ignored, because Image Services code compares the delete request object store (test) with the most recent federated object store (production). If they do not match, the delete request in Image Services is ignored, but Image Services returns a success to the object store that issued the delete request.

## Metadata mapping: Content Engine properties and Image Services indexes

Mapping metadata properties from Image Services to Content Engine is fairly straightforward, but limitations exist in certain cases. Table 3-1 on page 66 shows the correct data type mapping.

*Table 3-1 Data type mapping*

| Image Services data type | Display mask | Content Engine data type |
|---|---|---|
| Numeric | ######### (or fewer) | Float or integer |
| Numeric | ########## (or more) | Float |
| Numeric | #.## | Float |
| String | <*any*> | String (greater than or equal in size to Image Services string) |
| String | <*any*> | String (with choice list) |
| Date | <*any*> | Date or string |
| Menu | <*any*> | String (with choice list) |

## Data considerations

When planning, you need to consider multiple data types:

- Numeric

  Content Engine integers have a range of -2147483648 to 2147483647. If the Image Services numeric index has values outside of that range, use the float data type. Numeric data, which is longer than the mask allows, can exist in the index database. This situation can happen a variety of ways, including direct database manipulation. If you are in doubt, query the database for fields that are longer than a Content Engine integer property allows, or use a float data type.

  If the Image Services field is larger than a Content Engine float property, which is approximately 15 digits, the data will be federated, but with a loss of precision. If you are concerned about a loss of precision, you can use Image Services string indexes for extremely large numbers and map them to string properties in Content Engine.

  One downside of using float data types in Content Engine is that certain P8 applications, such as WorkplaceXT, display float data with a decimal, even if the decimal value is 0. Certain clients do not like the visual presentation of the data in this way. You will see examples of this presentation later in this chapter.

- String

  When mapping strings, be sure that the maximum size for the property in Content Engine is set at least as large as the mapped Image Services index. If you set it smaller than the corresponding value in Image Services, the

Content Engine will report an error and will not allow the document to be federated.

- Menu

  When creating a choice list in Content Engine that you map to an Image Services menu, specify values that correspond to the Image Services menu return codes, not the display value. Image Services stores the return code in the Image Services index database for the metadata value of the menu. We show examples of this situation later in this chapter.

  You can also map an Image Services string to a Content Engine string that has an assigned choice list, although this scenario is rare and risky. Certain Image Services clients want menu functionality (a limited set of allowable values), but they do not use Image Services menus. They create string indexes and use other processes to limit the allowable input values. On the Content Engine, they use choice lists. This scenario is possible, but the data in the Image Services string field must match what is allowed by the input processes. An Image Services user with SysAdmin privileges can update the string index with a value outside of the limits of the input processes. If this type of update happens, the field is mapped, and the value is not included in the choice list, the federation attempt fails. To prevent this situation, verify that all values for this field in the Image Services database are included in the Content Engine choice list.

- Date

  IS supports a wider range of dates than Content Engine. If you attempt to federate dates prior to 31 December 1752 or 12/31/1752, the federation will fail. We recommend that you perform a query of your index database for dates prior to 31 December 1752 and adjust them as appropriate before starting the federation.

  Typically, you map an Image Services date index to a Content Engine Date/Time property, but you can also map to a Content Engine string property. The data is stored in Content Engine in the *yyyy-mm-dd* format.

- System Image Services field mapping

  Clients often want to map several of the Image Services system indexes to Content Engine properties, because the information can be useful in searching for documents in Content Engine. The two most common indexes are f_docnumber and f_entrydate. You can map these indexes and other indexes, but f_docnumber can have a maximum value of 4294967295. This value is larger than the Content Engine integer data type can hold (2147483647), so if your document IDs are greater than 2147483647 or might be in the future, map the index to a float data type in Content Engine.

### Other mapping considerations

Also, consider these aspects of mapping:

- There is a 2K limit to the number of bytes that can be federated for a single document's metadata. Be sure that the data that you are mapping will not exceed this limit.
- Do not make Content Engine properties required during federation unless you verify that a value exists for every document in Image Services. If you attempt to federate a required property for which there is no corresponding Image Services value, the federation fails.
- Do not make Content Engine properties read-only during federation. This concept might seem obvious, but it has caused issues in the past. CFS cannot populate Content Engine properties with federated data if the property is set to read-only.
- If you are planning a large federation project, consider waiting until after the bulk of the federation is complete before creating any Content Engine property indexes (retrieval keys) on the mapped properties. Indexes slow the performance of the bulk import process. We discuss bulk import in more detail later in this chapter.

### Content-based retrieval

If the Content Engine is configured for full text content searching or *content-based retrieval* (CBR), users can search the content of text-based federated Image Services documents, such as `.txt`, `.rtf`, and `.doc` documents. Full-text retrieval of text-based Image Services documents requires P8 4.0 or later and Image Services 4.0 service pack 5 or later.

### Closed Image Services documents

By default, closed documents are not federated. Even if you export all documents for a certain document class, Image Services will not include closed documents in the CFS export. Many clients prefer this behavior, but you might want closed documents included in your federation. Later in this chapter, we include the steps to include closed documents.

### Versioning

If you allow versioning (check-out and check-in) on the federated document class, you can create a new version of a federated document. However, the new version will be treated as though it is a native Content Engine document. And similar to native Content Engine documents, annotations that are associated with the earlier version are not carried forward to the new version. You can prevent the ability to create versions using P8 APIs and tools by using FileNet Enterprise Manager to clear the selection from the Support Versioning field in the document class properties.

### Business process management

You can initiate workflow from federated documents, but we recommend that you do not enable this function during a bulk import of documents, because it can affect the speed of the federation. Wait until after the bulk import completes before enabling workflow on federated documents.

### Post federation migration

Sometimes after federating Image Services documents, you might want to migrate the content to a local filestore or another Content Engine fixed content device. IBM offers services to perform this migration for you. During and after the process, all of your federated document references remain intact; only the content is moved.

## 3.4 Installation

We do not include software installation steps as part of this book. Refer to the product documentation for FileNet P8 Platform for installation for procedures:

http://www.ibm.com/support/docview.wss?rs=3273&uid=swg27010422

## 3.5 Configuration

We used three systems to demonstrate the configuration of CFS-IS:

- Content Engine server

  We used the Content Engine to perform all of the steps on the P8 side. The actual tools that we used are FileNet Enterprise Manager and WorkplaceXT. You can run these tools from other workstations, but the examples that we include are run from the Content Engine server.

- Image Services (IS) server

  We used the Image Services server to create the document class, media family, and index objects using Application Executive (Xapex) and to demonstrate the Image Services logs and database tables containing CFS work queue items.

- Remote Administration Console tool (RAC) workstation

  We used a separate Windows system to run the Image Services export tool using RAC and to add, view, and update native Image Services documents using Integrated Desktop Management.

Archived

We moved from system to system to perform and test the configuration and included windows showing the steps as we execute each step. Several of the Image Services steps, such as the security administration and document class, family, and index configuration, can be done either from the Image Services server using Xapex or from the RAC workstation using RAC. Other steps can only be done from either the Image Services server using Xapex or from the RAC workstation using RAC. The Image Services export tool is only available from RAC.

### 3.5.1 All Windows servers: TCP/IP parameters

**Note:** This section applies to both Image Services and Content Engine Windows servers. It does not apply to UNIX servers.

System resource problems can be caused by a lack of available temporary ports or the amount of time that the server waits before reusing a closed socket ID. This section describes how to increase these parameters in the system registry. We recommend making changes on any Windows Image Services and Content Engine systems that are part of a CFS-IS configuration for these parameters:

- **MaxUserPort:** This setting determines the number of temporary ports that can be assigned on the server. When network traffic is extremely heavy, the server can run out of temporary ports unless you increase the MaxUserPort setting.
- **TcpTimedWaitDelay:** This setting determines the length of time that the server waits before reusing a closed ID socket. Although the default value is typically approximately 240 seconds, this parameter can safely be reduced to as few as 30 seconds.
- **TcpNumConnections:** This setting limits the maximum number of connections that TCP can have open concurrently. If this parameter is not defined, the default value of 6,777,214 (0xFFFFFE) is used. We recommend using the default value.

Archived

Follow these steps to verify or change the settings:

1. Open the registry editor:

   **Start** → **Run** → **REGEDT32**

2. Open **HKEY_Local_Machine** and drill down:

   **SYSTEM** → **CurrentControlSet** → **Services** → **Tcpip** → **Parameters**:

   – MaxUserPort

     If the parameter MaxUserPort exists, perform these steps:

     i. Double-click it to open it for editing.

     ii. If it is less than 65534 (decimal) or FFFE (hex), change the value to `65534` (decimal) or `FFFE` (hex).

     iii. Click **OK**.

     If the parameter MaxUserPort does not exist, create it:

     i. Right-click **Parameters**, and then, click **New** → ***DWORD value***.

     ii. Type `MaxUserPort` in the Name field.

     iii. Double-click it to open it for editing.

     iv. Change the value to `65534` (decimal) or `FFFE` (hex).

     v. Click **OK**.

   – TcpTimedWaitDelay

     If the parameter TcpTimedWaitDelay exists, perform these steps:

     i. Double-click it to open it for editing.

     ii. If it is greater than 30 (decimal) or 1E (hex), change the value to `30` (decimal) or `1E` (hex).

     iii. Click **OK**.

     If the parameter TcpTimedWaitDelay does not exist, create it:

     i. Right-click **Parameters**, and then, click **New** → ***DWORD value***.

     ii. Type `TcpTimedWaitDelay` in the Name field.

     iii. Double-click it to open it for editing.

     iv. Change the value to `30` (decimal) or `1E` (hex).

     v. Click **OK**.

Archived

– TcpNumConnections

If the parameter TcpNumConnections exists, follow these steps:

i. Delete the parameter to set it to the default.

If the parameter TcpNumConnections does not exist, do not create it.

3. Restart the server for the changes to take effect.

## 3.5.2 Image Services

Perform user administration and database maintenance, which we describe in this section, on the Image Services server.

### User administration

Create the user, and adjust it appropriately with the following steps:

1. Create a CFS-IS administrator user.

   The Content Engine Import Agent performs the following tasks:

   a. Log in to Image Services as this user to import and update document metadata that was exported by Image Services.

   b. Delete federated documents from Image Services when they are deleted from Content Engine.

   c. Update security when documents are declared as records in Content Engine.

   If more than one Content Engine will access the same Image Services server, we recommend creating a separate Image Services user name for each Content Engine system that will connect to the Image Services server.

   The CFS user needs to be a member of a group or groups that will allow read, write, and append/execute privileges to the document classes and the documents that will be federated.

**CFS-IS user ID naming restrictions**: The Image Services user name that Content Engine uses for CFS-IS connectivity must not contain any of the following special characters, because these characters are either not valid in Content Engine or not valid in Image Services:

_ # [ ] ! < > ? *

Archived

The Image Services user must be separate from ordinary users or system administrator users. For our example, we used the name CFSAdmin and assigned it to the primary group SysAdminG, as shown in Figure 3-2.

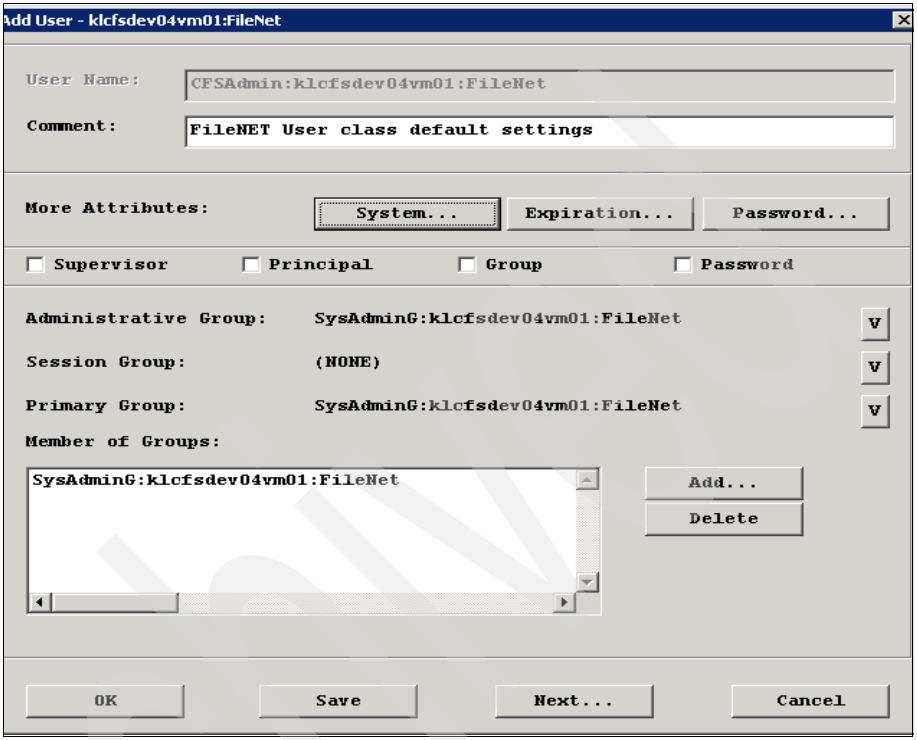



*Figure 3-2 Create CFS-IS administrator user*

2. Set the password for the CFS-IS administrator user, as shown in Figure 3-3 on page 74. We use the password `CFSAdmin`.

*Figure 3-3 Set CFS-IS user password*

3. The CFS-IS user will need, at a minimum, three concurrent logins for each object store that is configured for CFS-IS (plus three more logins if federating annotations). By default, Image Services only allows one concurrent login for each user, so it is necessary to increase the number of allowable concurrent logins, as we have done in Figure 3-4.

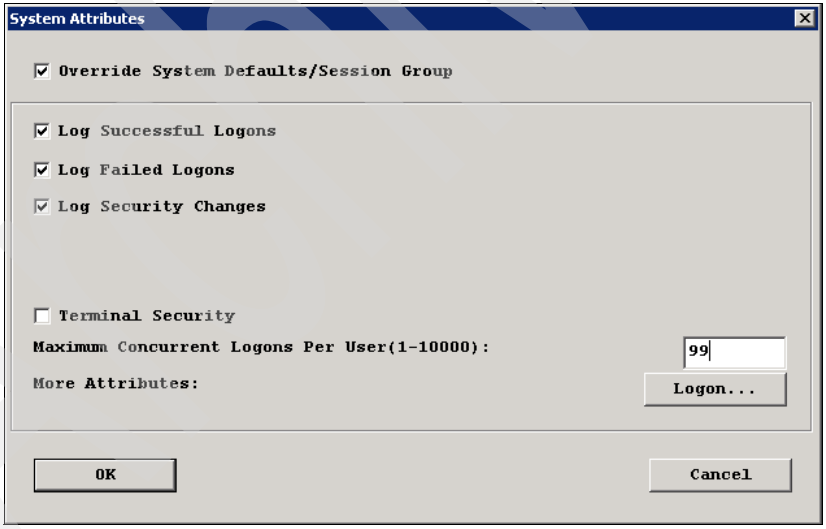



*Figure 3-4 Increase allowable concurrent logins for CFS-IS user*

## Database maintenance

Create a media family, user indexes, and a document class that will be used for federation, or use a media family, user indexes, and a document class that exist already. We have created a new a media family, user indexes, and a document class using the following steps:

1. IS media family

   We have created a new family named `MSAR2G`, as shown in Figure 3-5.

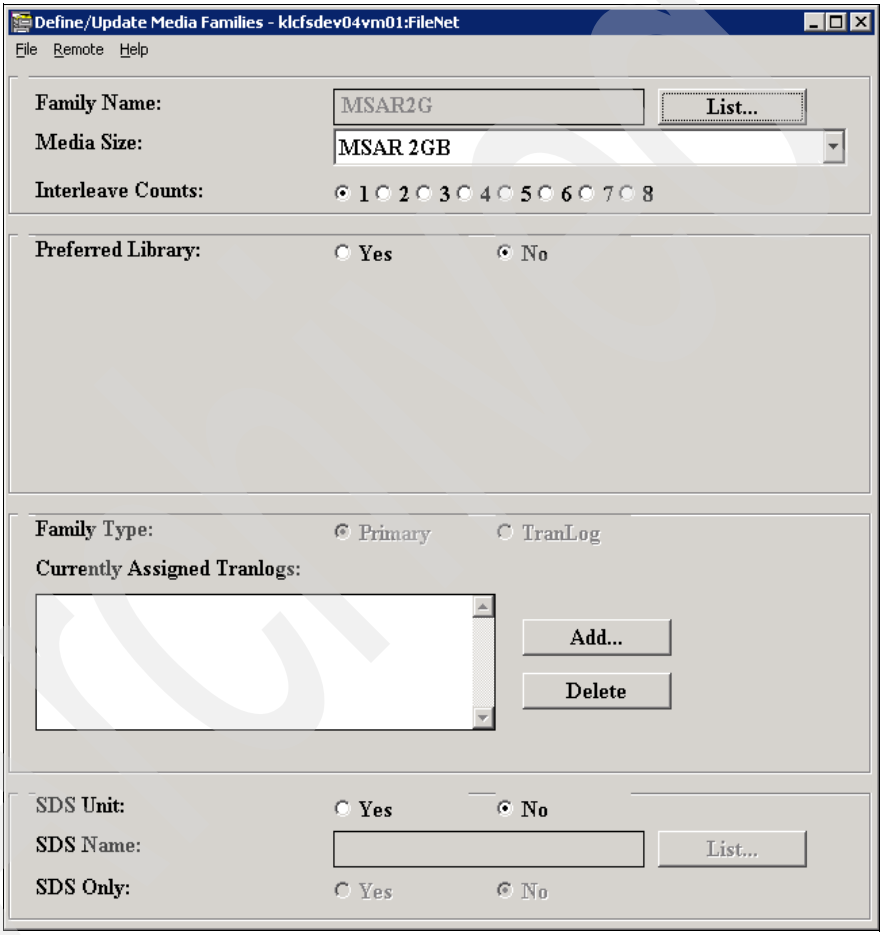



*Figure 3-5 Create a media family*

2. User indexes

   We have created an index for each data type (numeric, string, date, and menu):

– AccountNumber (Figure 3-6)

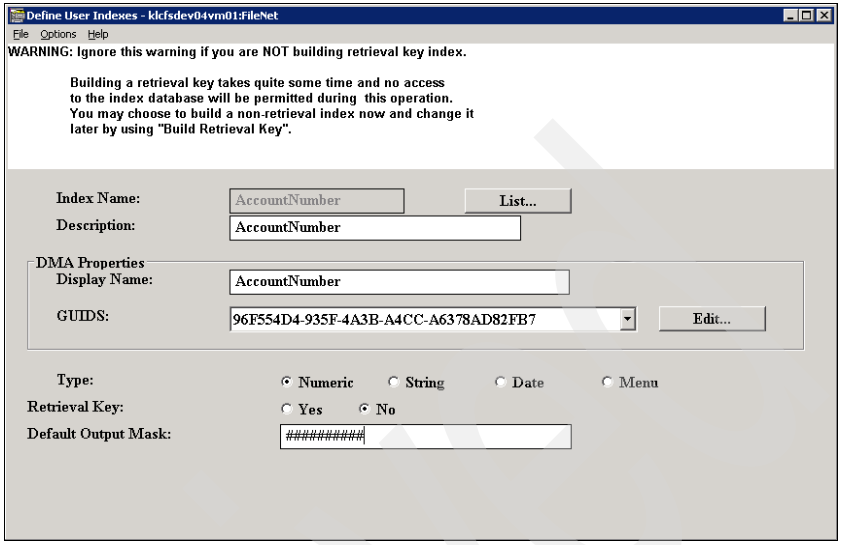



*Figure 3-6 Numeric index*

– LastName (Figure 3-7)

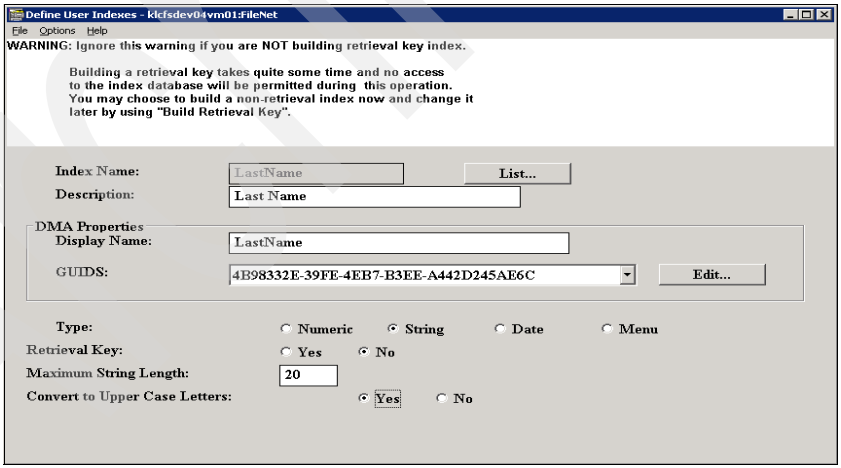



*Figure 3-7 String index*

– PolicyDate (Figure 3-8 on page 77)

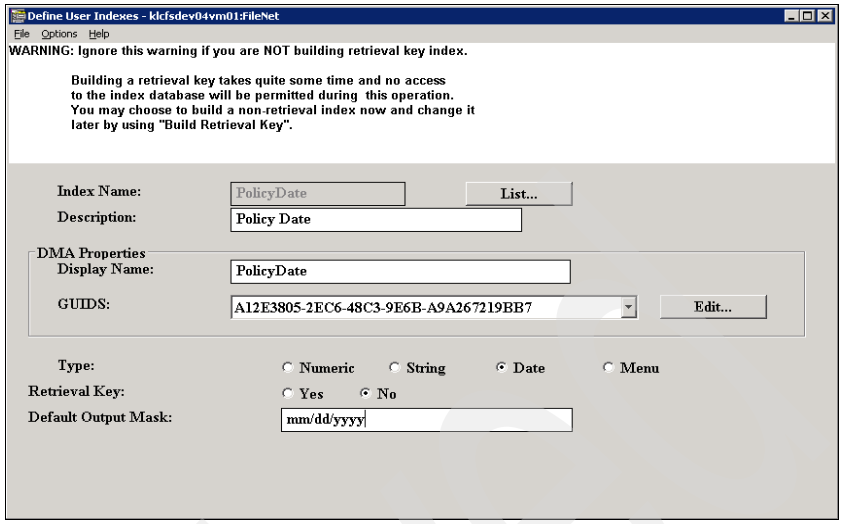



*Figure 3-8 Date index*

– State (Figure 3-9)

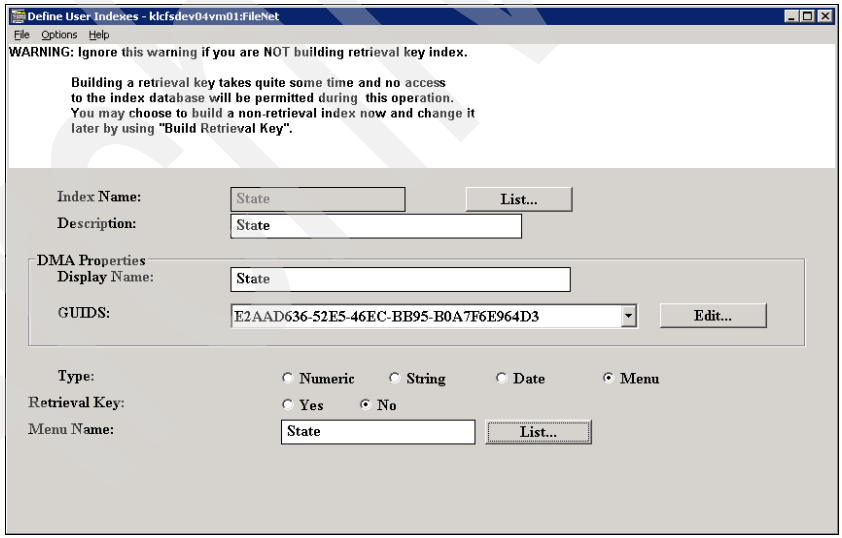



*Figure 3-9 State index*

Archive

– State menu items (Figure 3-10)

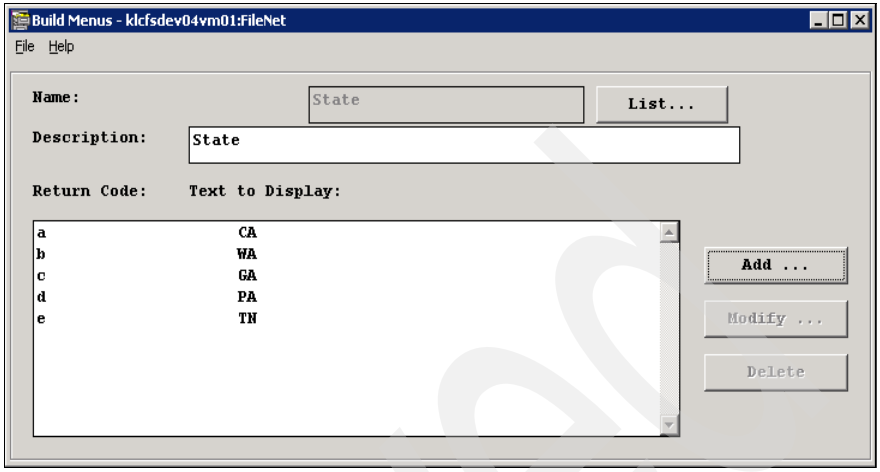



*Figure 3-10 State menu items*

3. Document class

   In this example, we have created a new class named RbDC, assigned it to the MSAR2G media family, and assigned the previous indexes to the document class. See Figure 3-11 on page 79 and Figure 3-12 on page 79.

   You can configure CFS-IS in multiple ways:

   – If you want to be able to retrieve documents only from the Content Engine server, set Enable Cataloging to **No**. Documents that are added to the document class will be federated and the metadata will be copied to the Content Engine, but there will be no record of the document stored in the Image Services index database after the import is complete. Therefore, you will not be able to retrieve the document, or even see the document reference with a query, using Image Services applications, such as Integrated Desktop Management.

   – If you want to be able to retrieve the documents from both Image Services and P8 applications, set Enable Cataloging to **Yes**. Document references and metadata will remain in the Image Services index database after federation. For our example, we have configured our system in this manner.

   – You can enable cataloging for a test period and later disable it and re-export the documents by selecting **Delete after export**. The document references will be removed from the Image Services index database.

Archived

**Important:** Always enable cataloging for documents that are stored using NLS, because NLS requires cataloging.

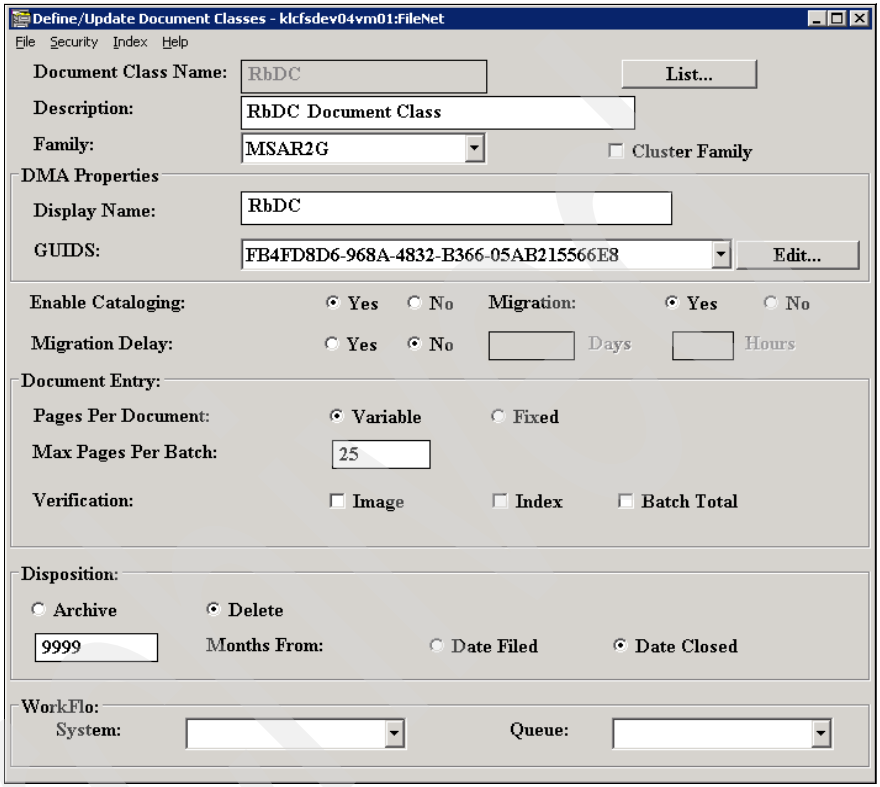



*Figure 3-11 Document class setup*

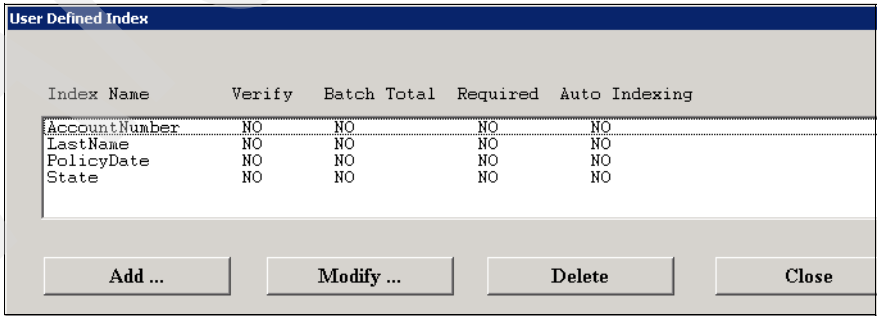



*Figure 3-12 Indexes assigned to document class*

## 3.5.3 Content Engine

Content Engine configuration consists of the following steps:

1. Verify the network prerequisites.
2. Create an object store.
3. Create a fixed content device.
4. Create a fixed storage area.
5. Create Content Engine document objects.
6. Enable federation in Image Services.

### Verify the network prerequisites

In the first steps of the CFS-IS configuration on the Content Engine, you need network, security, document class, and user index information from the Image Services system.

Determine the following information about the Image Services system:

- IP address of the Image Services server

  Verify that you can ping the IP address from the Content Engine server.
- IS domain and organization name

  The domain name is sometimes set to the same name as the root/index servername, but it is not always the same.
- Server name of the Image Services root/index server
- CFS-IS admin user name and password
- IS document classes and user indexes to be mapped

For our example, Table 3-2 on page 81 shows the system information.

Archived

*Table 3-2 IS system information*

| IS system setting | Valued used in our example |
|---|---|
| IP address of Image Services root/index server | 192.168.10.103 |
| Server name of Image Services root/index server | klcfsdev04vm01 |
| Domain and organization of Image Services system | klcfsdev04vm01:FileNet |
| CFS-IS admin username | CFSAdmin |
| CFS-IS admin password | CFSAdmin |
| IS document class | RbDC |
| Numeric Image Services user index | AccountNumber |
| String Image Services user index | LastName |
| Date Image Services user index | PolicyDate |
| Menu Image Services user index | State |

### *Four-part Network Clearing House name*

The Content Engine communicates to the Image Services using the four-part Network Clearing House (NCH) name of the Image Services. You determine this name using the following steps:

1. Start with the Image Services domain and organization.
2. Convert all uppercase letters to lowercase.
3. Remove all characters, except ASCII alphanumeric characters and hyphens.
4. Insert a hyphen between the domain and organization.
5. Append `-nch-server` to the end of the string.

Table 3-3 shows a few examples.

*Table 3-3 NCH name examples*

| IS domain:organization | Four-part NCH name |
|---|---|
| klcfsdev04vm01:FileNet | klcfsdev04vm01-filenet-nch-server |
| IMS1:IBM | ims1-ibm-nch-server |
| IS_mortgage:FileNET | ismortgage-filenet-nch-server |

When you have determined the four-part NCH name, you need to set up your network or Content Engine server to use the correct IP address for the four-part name. You can either add this name/IP address to your Name Resolution Service

(Domain Name System (DNS) and Network Information Service (NIS)) or to the Content Engine `hosts` file.

### *Using a local hosts file*

If you use the Content Engine `hosts` file, use the correct format:

*{ip-address}  {IS root/index servername}  {four-part NCH name}*

The `hosts` file is typically in these locations:

**UNIX**: `/etc/hosts`

**Windows**: `C:\windows\system32\drivers\etc\hosts`

In our example, the `hosts` file has the following entry:

```
192.168.10.103    klcfsdev04vm01    klcfsdev04vm01-filenet-nch-server
```

### *Using a name resolution server*

If you prefer to use a name resolution service, such as DNS, an alias entry for the Image Services root/index server must include the four-part NCH name, for example:

- IS domain/organization: `IMS_1:FileNet`
- IS root/index server name: `isprod01`

Add an alias to the entry for server isprod01:

Alias name=`ims1-filenet-nch-server`

### *Verify connectivity*

From the Content Engine server, use the ping and nslookup tools to verify communication to the Image Services root/index server name, IP Address, and four-part NCH name.

## Create an object store

You can create a fixed content device before creating the object store. However, for CFS-IS, this sequence can lead to problems, so we recommend creating the object store first. We do not describe the steps to create an object store first in this book. For more information about how to create an object store, see the FileNet P8 product documentation.

Archived

## Create a fixed content device

The Content Engine connects to external storage devices and systems using a fixed storage area. The connection is provided by a Content Engine object called a *fixed content device* (FCD). Each type of FCD communicates using an API that is specific to the external system, such as IBM FileNet Image Services, IBM Content Manager OnDemand, IBM Tivoli® Storage Manager, EMC Centera, NetApp Snaplock, and other federated devices using IBM Content Integrator. CFS-IS uses a fixed content device type for communicating to Image Services, which is the type that we created in the following steps:

1. Launch FileNet Enterprise Manager Administration Tool. Select **Start** → **Programs** → **IBM FileNet P8 Platform** → **FileNet Enterprise Manager Administration Tool**.

   **Tip:** There are several ways to start many of the task wizards within FileNet Enterprise Manager. For example, to create a new fixed content device, perform one of these tasks:

   - Right-click **Fixed Content Devices**, and select **New Fixed Content Device**.
   - Right-click **P8 domain**, and select **New Fixed Content Device**.

2. Log in to the Content Engine as a user with administrator privileges.
3. Right-click **Fixed Content Devices**, and select **New Fixed Content Device**.
4. Assign a name to the FCD. Choose a name that indicates the Image Services system to which you will connect. Click **Next**.

   **Note:** By default, the FCD name will be added to the beginning of the document title of the documents that are federated from Image Services.

5. From the Type list at the top of the window, choose **IBM FileNet Image Services**. For the configuration parameters, you can accept the default values, except for these five items, which are shown with our example values:

| | |
|---|---|
| IS User Name | `CFSAdmin` |
| IS Password | `CFSAdmin` |
| IS Domain Name | `klcfsdev04vm01` |
| IS Organization | `FileNet` |
| Timeout Seconds | `60` |

**Important:** Do not use the Tab key to move to the next field, because the Tab key cancels the wizard. You can use the up and down arrow keys instead.

See Figure 3-13 for our example.

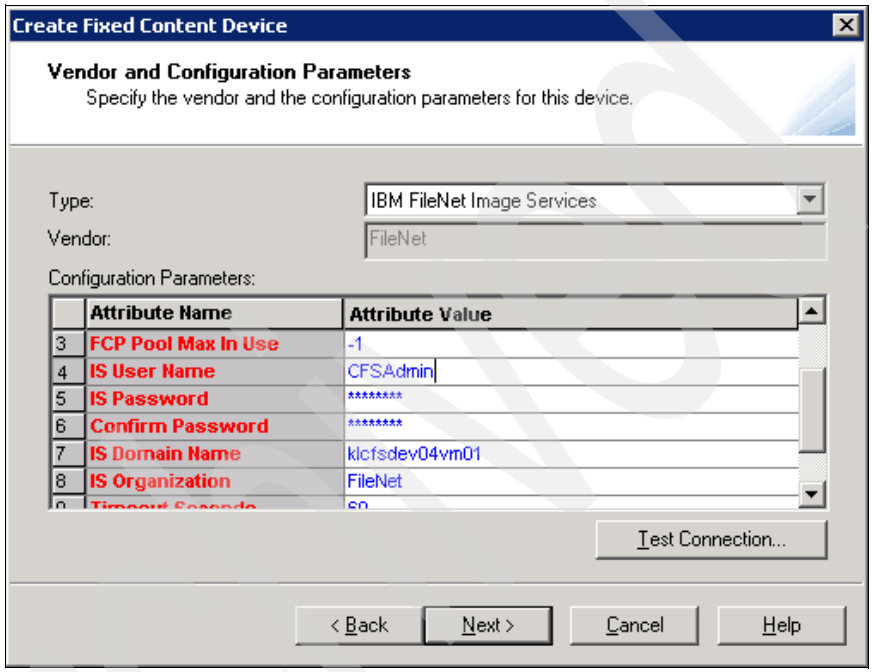



*Figure 3-13 Fixed Content Device parameters*

6. Click **Test Connection**, which will perform a test login to the specified Image Services system with the specified login name and password. If successful, you see a confirmation (Figure 3-14).

*Figure 3-14 Fixed Content Device: Test Connection successful*

7. Click **Next**.

8. When you are prompted to select a default Image Services document class, you must select one of the Image Services document classes that are displayed. For our use case, it does not matter which Image Services document class you choose. This selection is only important for the use case where the content for documents that are added through P8 is stored in Image Services. Even in that case, you can override the default Image Services document class in the fixed storage area parameters. After selecting a document class, click **Next**.
9. When you are prompted for site-specific settings, click **Next** unless you want to improve the performance of retrievals from remote locations. With Image Services, you can create Cache Services Manager caches at the remote site and add the Cache Services Manager cache name here. We discuss remote caching in more detail later in this chapter.
10. Click through the rest of the wizard until it is finished and you receive a success message, such as in Figure 3-15.

*Figure 3-15 Successful fixed content device creation*

## Create a fixed storage area

We now create a fixed storage area that uses the FCD that we created in the previous steps.

> **For more information:** Unlike fixed content devices, which are available to the entire P8 system, a fixed storage area is specific to a single object store. For more information about fixed storage areas, click **Help on FileNet Enterprise Manager Administration tool** → **Content Storage** → **Fixed Storage Areas**.

Follow these steps to create a fixed storage area:

1. Right-click the name of the object store that contains, or will contain, the Content Engine document class that you want to map to the Image Services document class.
2. Click **New** → **Storage Area**.
3. Click **Next** at the welcome panel.
4. When prompted to select a site, click **Next** for Initial Site.

5. Assign a name to the storage area, and click **Next**. We enter the name `IS412_sa`.
6. For storage area type, select **Fixed Storage Area**, and click **Next**.
7. For Fixed Content Device, select the FCD that was created in the previous step.
8. The next window prompts you for the following parameters:
   – For the Staging Area Path, specify an existing folder name on the Content Engine. This location is used to store content temporarily (in the use case where the content of documents, which are added from P8, is stored in Image Services). This location is also used to store new versions of the federated document that are created by P8 applications. These new versions are no longer considered federated, but they are native P8 documents.
   – For the IS Document Class, leave Use default selected, and click **Next**. Or, you can choose another Image Services document class, but for our use case, it makes no difference.
9. For the Storage Area Size and Delete Method, leave the defaults or specify whatever values you prefer, and click **Next**.
10. The next window prompts you for a storage policy.

> **For more information:** A *storage policy* provides mapping to a single or multiple storage areas for a Content Engine object that has content, such as a document. If an object, such as a document class, has a specified (non-default) storage area and a specified storage policy set, the storage area takes precedence.
>
> For more information about storage policies, click **Help on FileNet Enterprise Manager Administration tool** → **Content Storage** → **Storage Policies**.

If you have already created a storage policy for CFS-IS, select it in the “Map storage area to existing Storage Policies” box. If not, select **Create Storage Policy**, assign it a name, and click **Next**. We choose the name `IS412_sa_sp`.

11. Click **Finish** to complete the creation of your storage area. You get a message indicating that the storage area and storage policy were created successfully, as shown in Figure 3-16 on page 87.

*Figure 3-16 Fixed storage area and storage policy creation successful*

## Create Content Engine document objects

Before we can federate Image Services documents, we must first create a Content Engine document class to map to the desired Image Services document class and Content Engine property templates to map to the desired Image Services indexes. We must assign the Content Engine property templates to the Content Engine document class. Use these steps to create the Content Engine document objects:

1. Create property templates.
2. Create file storage area and storage policy.
3. Create document class.
4. Create mapping between Image Services and Content Engine document classes.
5. Enable Image Services Import Agents.

### *Create property templates*

*Property templates* are metadata objects that correspond to Image Services user indexes, such as LastName and PolicyDate. In our example, we created the property templates that are displayed in Table 3-4.

*Table 3-4 Property templates*

| Property template name | Data type |
|---|---|
| Last Name CFSIS | String |
| State CFSIS | String with choice list |
| Policy Date CFSIS | DateTime |
| Policy Number CFSIS | Integer |
| IS Docnumber CFSIS | Float |
| IS Entrydate CFSIS | DateTime |
| IS Number of Pages CFSIS | Integer |

Archived

We include detailed step-by-step examples of the creation of the first two templates in the following steps.

Create the property template, Last Name CFSIS:

1. In IBM FileNet Enterprise Manager, expand the **Object Stores** node.
2. Right-click the desired object store, and select **New** → **Property Template**.
3. Click **Next** on the welcome window.
4. Assign a name to the property template, and click **Next**. In this example, we used the name `Last Name CFSIS`.

> **Clarification:** The mapped property template and the document class names can contain spaces. They do not have to be the same name as the corresponding Image Services object (but it is a good practice if the names are similar). For example, a Content Engine property template named Account Number can map to an Image Services index named accountNum. Also, in our examples, we have appended `CFSIS` to the name, but this approach is not required.

5. Select the data type. Last Name CFSIS is a string data type.
6. For the choice list and marking set selections, keep the defaults (both selections are set to none), and click **Next**.
7. When prompted for Single or Multi Value, select the default of Single.
8. To increase or decrease the maximum size, click **More** and change Maximum String Length. Mapped property templates must be greater than or equal to the size of the corresponding Image Services index value. Click **OK**, **Next**, and then, **Finish** to complete the process.

Create the property template, State CFSIS:

1. In IBM FileNet Enterprise Manager, expand the **Object Stores** node.
2. Right-click the desired object store, and select **New** → **Property Template**.
3. Click **Next** on the welcome window.
4. Assign a name to the property template, and click **Next**. In this example, we used the name `State CFSIS`.
5. Select the data type. State CFSIS is a string data type.
6. When prompted for a choice list assignment, if you have already created it, select it from the drop-down list, or click **New** to create a choice list assignment.

**Note:** *Choice lists* are comparable to Image Services menus. In both cases, they limit the metadata to a predefined set of allowable values. For example, you might want a property template for the two-letter abbreviation for US State, and you might not want the users to manually enter the value, because they might enter an invalid value, such as `MM`. When a choice list is assigned to a property template, the user is only able to select a value from a drop-down list of items in the choice list. Choice lists can be shared among multiple property templates. For example, you can create a choice list that contains only “Yes” and “No”. This choice list can be assigned to two separate string property templates, such as “Contract Closed” and “Signature Obtained”.

Follow these steps to create a new choice list:

a. At the Choice list wizard welcome window, click **Next**.

b. Assign a name to the choice list. We use the name `State Choice List CFSIS`.

c. When prompted for the data type (String or Integer), select the choice that corresponds to the data that will be mapped. Our example is a string data type.

d. When prompted for choice list elements, add the list of desired items by clicking **New Items**. You can also create logical groups of items by clicking New Groups. When you create a new item, you first type the display name. The value will be set automatically to the same value as the display name, without spaces, but you can manually change the value without affecting the display name. The user sees the display name in P8 applications, but the value is what is stored in the database and the value must match the data that will be federated from Image Services. See Figure 3-17 on page 90 for our example.

Archived

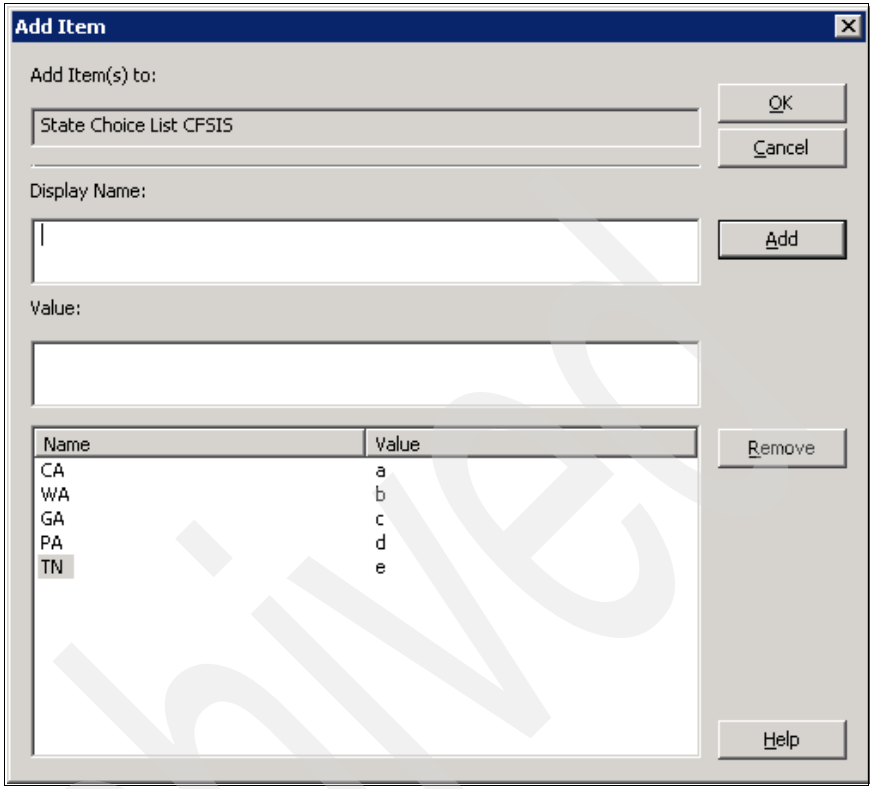



*Figure 3-17 Choice list: Add Item window*

e. When you have added each item, click **Next** and **Finish**. You will see a message that the choice list was created successfully.

f. You will be returned to the property template wizard. Verify that the new choice list is assigned in the drop-down list. Click **Next**.

g. When prompted for Single or Multi Value, select the default of Single.

7. To increased or decrease the maximum size, click **More** and change Maximum String Length. Mapped properties must be greater than or equal to the size of the corresponding Image Services index value. Click **OK**, **Next**, and then, **Finish** to complete the process.

Follow the same steps to create the remaining property templates.

### *Create file storage area and storage policy*

We now create a file storage area and storage policy for the local storage of documents that are added through P8 and whose content will not be stored in Image Services. Creating this file storage area and policy is optional for CFS-IS,

because you can use the default database storage area and policy. For more information about comparing file storage and database storage, click **Help on FileNet Enterprise Manager Administration tool** → **Content Storage** → **File Storage Areas and Database Storage Areas**.

Follow these steps to create the file storage area and storage policy:

1. In IBM FileNet Enterprise Manager, expand the **Object Stores** node.
2. Right-click the desired object store, and select **New** → **Storage Area**.
3. Click **Next** on the welcome window.
4. When prompted to select a site, click **Next** for Initial Site.
5. Assign a name to the storage area and click **Next**. We choose the name `FileStorageArea`.
6. For storage area type, select **File Storage Area**, and click **Next**.
7. When prompted for Shared Directory, type or browse to an existing path that is not used by another file storage area.
8. When prompted for storage area size, select the parameters that most closely meet your needs, and click **Next**. We accept the default settings.
9. When prompted to select a storage policy, you can use an existing storage policy or create a new storage policy. Click **Next** and **Finish** to complete the process. You will get a message that the storage area was created successfully.

> **Note:** Do not use the default database storage policy or the storage policy that is associated with the fixed storage area that was created earlier. We select the choice to create a new storage policy and name it `FileStorageArea_sp`.

### *Create document class*

We now create a document class that will contain the federated documents from the Image Services document class named RbDC. We will also assign the property templates that were previously created to the new class.

> **Note:** With Content Engine, you can create a hierarchical document class structure with classes and subclasses. Subclasses inherit security, properties, and other settings from its parent class. You can modify several of the inherited values in an individual subclass, if desired. For more information about document class inheritance, see Help on **FileNet Enterprise Manager Administration tool** → **Classes** → **Concepts** → **Inheritance**.

Archived

Follow these steps to create the document class:

1. In IBM FileNet Enterprise Manager, expand the **Object Stores** node.
2. Right-click the desired object store, and select **New** → **Class**.
3. Click **Next** on the welcome window.
4. Assign a name to the document class, and click **Next**. In this example, we used the name `Rb_dc_CFSIS`.
5. When prompted to select a superclass, select **Document** or whatever parent class you prefer, and click **Next**.
6. When prompted to select properties, select each of the properties that you created in the previous step, and click **Add**. When you have added all of the desired properties, click **Next**.

   **Note 1:** If you have previously created a parent class with assigned properties that you want to propagate to this class, click **Show Inherited** to display the inherited properties.

   **Note 2:** If you need to assign a property template that does not yet exist, click **New**. You will enter the wizard to create the new property and, then, return to this window when creating the property is complete.

7. When you are prompted to the set class level property attributes, click **Next**.
8. When you are prompted for the content storage parameters, select the best choice for your environment, and click **Next**. In our example, we selected the storage policy named **FileStorageArea_sp** that we created in the previous step.

   **Note:** This selection only applies to documents that are added through P8. If you select the fixed storage area or policy that points to the fixed storage area that you created earlier, the content of documents that are added through P8 will be stored in Image Services.

9. When you are prompted to configure auditing, click **Next** and **Finish** to complete the process. You will get a confirmation that the class was created successfully.

Archived

### *Create mapping between Image Services and Content Engine document classes*

Perform these steps to create the mapping between the Image Services and Content Engine document classes:

1. In IBM FileNet Enterprise Manager, expand the **Object Stores** node.
2. Right-click the desired object store, and click **Properties**.
3. Select the **Content Federation** tab.
4. Click **Add**.
5. From the Fixed Content Device drop-down list, select the FCD that you created earlier.
6. From the Available External Classes drop-down list, select the Image Services document class that you created earlier, and click **OK**. See Figure 3-18.

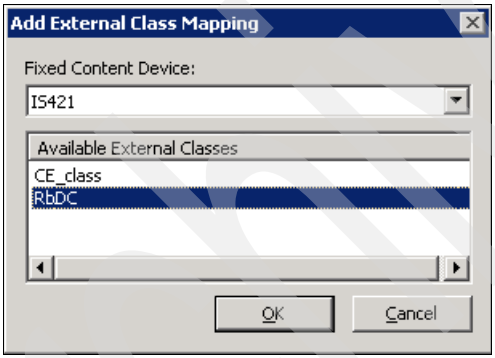



*Figure 3-18 Add external class mapping window*

7. From the External Repository Properties on the left side of the window, select one of the Image Services indexes that you want to map. Only Content Engine properties that have mappable data types will be displayed on the right side of the window. Click the corresponding Content Engine property, and click **Add**. The mapping will be displayed on the bottom pane in the Current Property Mappings window. See Figure 3-19 on page 94.

Archived

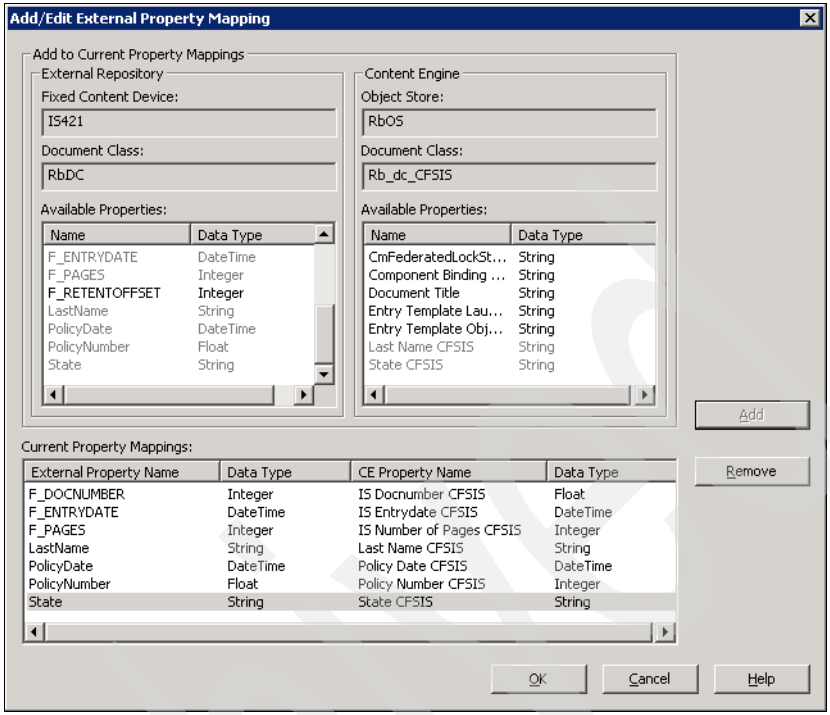



*Figure 3-19 Property mapping window*

8. When you have mapped each of your desired properties, click **OK**. You can edit the mapping by clicking the configured mapping and clicking **Map Properties**. See Figure 3-20.

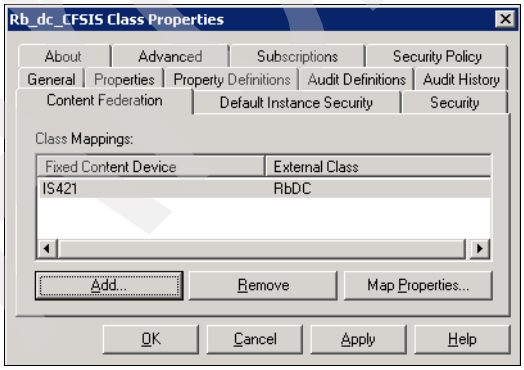



*Figure 3-20 Content Federation tab*

9. Click **OK** to exit the Properties window.

### *Enable Image Services Import Agents*

The configuration of CFS-IS on the Content Engine system is complete. We still have another step to perform in Image Services, but we can enable the Import Agents in Content Engine now:

1. In IBM FileNet Enterprise Manager, right-click the top level P8 domain, and click **Properties**.
2. Select the **IS Import Agent** tab. See Figure 3-21 for the default values of the Image Services Import Agent.

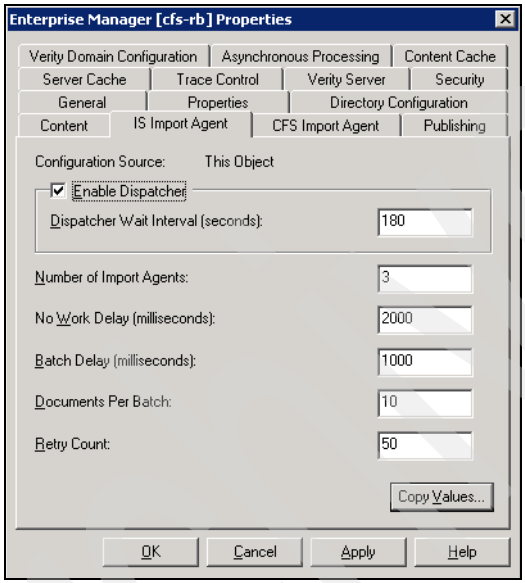



*Figure 3-21 IS Import Agent tab*

You can adjust the parameters to tune the performance of your system. See Table 3-5 on page 96 for a description and range for each parameter.

Archived

*Table 3-5 Image Services Import Agent parameters*

| Parameter | Description | Range |
|---|---|---|
| Dispatcher Wait Interval | Time in seconds before starting Import Agents | 5 - 600 |
| Number of Import Agents | Number of agents per Content Engine instance that connect to Image Services and perform work | 1 - 5 |
| No Work Delay | Time in milliseconds before looking for work after finding *no* work the last time that the Image Services Import Agent checked | 0 - 100000 |
| Batch Delay | Time in milliseconds before looking for work after finding work the last time that the Image Services Import Agent checked | 0 - 1000 |
| Documents Per Batch | Number of documents that the Import Agent will import in each batch | 1 - 500 |
| Retry Count | If the Import Agent cannot import a batch, it will retry the number of times that is specified. | 1 - 10000 |

**For more information:** For more guidance, see the *IBM FileNet P8 Performance Tuning Guide*, which you can download it from the following Web site:

http://www-01.ibm.com/support/docview.wss?rs=3247&uid=swg27010422

3. Select **Enable Dispatcher**, and click **OK**.

## Enable federation in Image Services

In Image Services, we have two primary tasks to start federating documents. To enable the federation of new documents, we must create a default map for the desired Image Services document class. To enable the federation of existing documents (IS documents that existed before the default mapping was created), we must export the documents. In both cases, we use the Image Services Remote Administration Console on the RAC workstation to perform the following tasks:

1. Enable the federation of new documents.
2. Enable the federation of existing documents.

### *Enable the federation of new documents*

Now, we configure a mapping from the Image Services document class to the Content Engine object store to which it will be federated. This mapping will enable federation for future documents that are committed to this document class.

The following steps describe the process:

1. Start the Remote Administration Console on the RAC workstation. Click **Start** → **Programs** → **FileNet** → **IS Remote Admin Console**.

   **Using RAC:** The process to map Image Services document classes to Content Engine object stores is only available using RAC. It is not available in the Image Services Application Executive tool (Xapex).

2. Log in as SysAdmin or as another member of the SysAdminG primary group.
3. Select **Applications** → **IS Catalog Export Tool**.
4. Select the **Default Document Class Mappings** tab.

   **Maps:** You can map an Image Services document class to multiple Content Engine document classes if each Content Engine document class is in a separate object store. However, there is only one *default* Image Services document class mapping. The default Image Services mapping controls which object store will contain the federated data when new documents are entered into Image Services. You can create and select additional maps to other object stores and, then, federate a previously federated range of documents to another object store using the non-default maps.

5. In the Content Engine Configured IS Doc Classes drop-down list, select the Image Services document class that you want to federate, which must be the same Image Services document class that you mapped in Content Engine in the previous steps. In our example, we selected **RbDC**.
6. Select the appropriate Content Engine domain and object store, which must be the same as the object store that you mapped in Content Engine in the previous steps. In our example, we selected **cfs-rb - RbOS**.
7. When both the IS document class and the object store are selected, click **Set Default Map**. See Figure 3-22 on page 98.

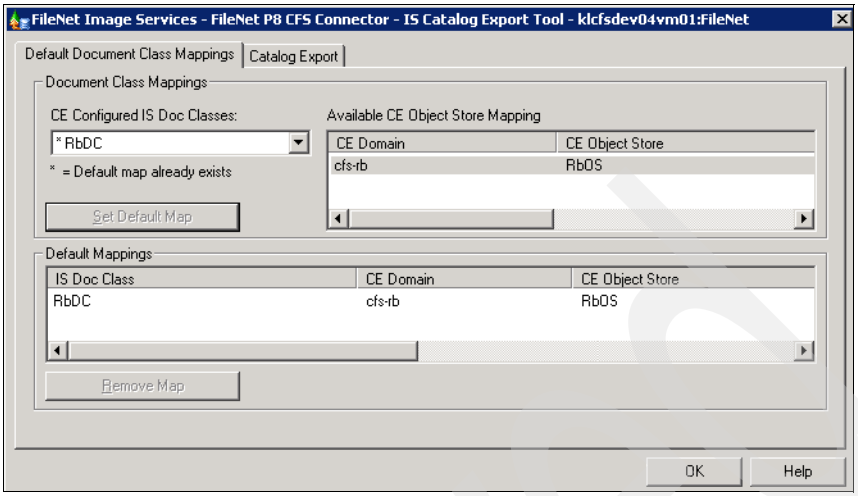



*Figure 3-22 IS document class default mapping*

8. Click **OK**.

Everything is now configured and enabled. New Image Services documents will be federated to Content Engine. After describing the process to federate existing documents, we will test the functionality.

### *Enable the federation of existing documents*

Perform these steps to enable the federation of existing documents:

1. Start Remote Admin Console on the RAC workstation. Click **Start** → **Programs** → **FileNet** → **IS Remote Admin Console.**

> **Using RAC:** The process to export existing Image Services documents for federation to Content Engine is only available using RAC. It is not available in the Image Services application executive tool (Xapex).

2. Log in as SysAdmin or another member of the SysAdminG primary group.
3. Select **Applications** → **IS Catalog Export Tool**.
4. Select the **Catalog Export** tab.
5. Select the following export parameters:

| | |
|---|---|
| **IS Doc Class** | Select the previously configured Image Services document class. |
| **CE Domain** | Select the previously configured CE domain. |

| | |
|---|---|
| **CE Object Store** | Select the previously configured CE object store. |
| **First Doc ID** | Select the first document ID in the desired range. |
| **Last Doc ID** | Select the last document ID in the desired range. |
| **Delete After Export** | If checked, the document catalog is removed after federation. |
| **Re-Export** | If checked, the document is re-exported even if it has already been exported. |
| **Annotations Only** | If checked, only document annotations will be exported. |

The typical setting is to leave the first and last doc ID at the default settings (100000 - 3999999999) and to *not* select any of the check boxes. This setting results in the export of all documents, in the specified class, that have not already been federated. This setting leaves the document in the index database and available for viewing with Image Services applications.

> **Important:** Only select Delete After Export if you are sure that you want to remove the catalog entry from the Image Services index database. You will no longer be able to view the document or see its metadata using Image Services applications, such as Integrated Desktop Management. Do not use this option for documents that are stored using NLS, because NLS requires cataloging.

6. Click **Add**.
7. Click **Export Now**. See Figure 3-23.

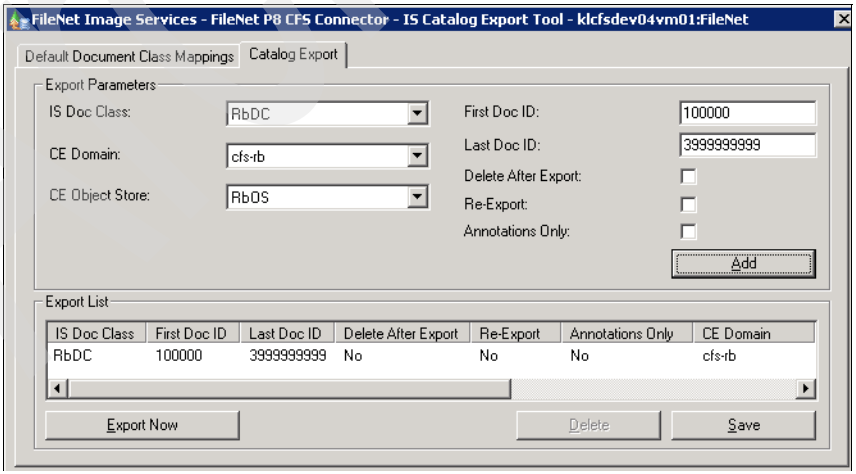



*Figure 3-23 Catalog export tool*

## 3.5.4 Testing Content Federation Services for Image Services

Now, all configuration steps are complete, and we can test adding new documents to Image Services and verifying that they are available in Content Engine:

1. Add the document to Image Services.
2. Find the document in Image Services.
3. Search from WorkplaceXT.

### Add the document to Image Services

You can add documents to Image Services using multiple methods, including FileNet Capture, HPII, and Integrated Desktop Management. For our example, we used Integrated Desktop Management. We used these steps:

1. Log in to a workstation with Integrated Desktop Management configured for your Image Services system.
2. From your desktop, open **FileNet Neighborhood**. See Figure 3-24.

*Figure 3-24 FileNet Neighborhood*

3. Drag a document onto your library. In our case, the name is **IS41**.
4. If prompted, log in as SysAdmin or another user that has permissions to add documents to the mapped document class, which, in our example, is RedbokDC.
5. Select the mapped document class and enter values for the indexes. See Figure 3-25 on page 101.

Archived

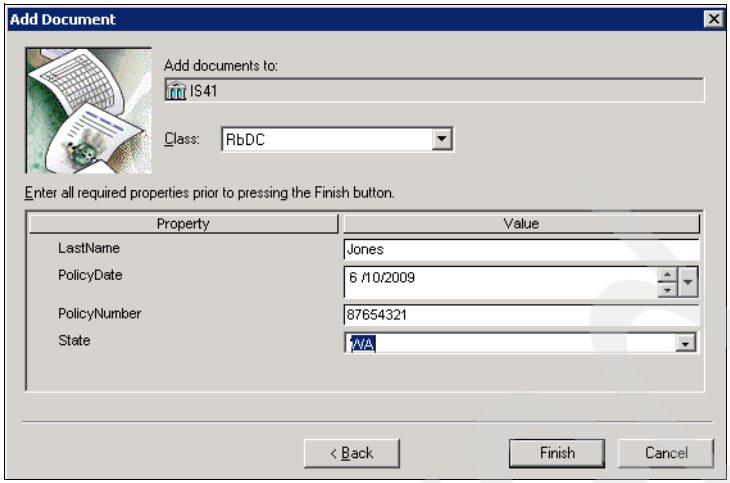



*Figure 3-25 IS document index values*

6. Click **Finish**. The new document ID is displayed briefly.

## Find the document in Image Services

To confirm that the document was successfully added to Image Services, follow these steps:

1. Log in to a workstation with Integrated Desktop Management configured for your Image Services system.
2. From your desktop, open **FileNet Neighborhood**.
3. Right-click the library, and select **Find**.
4. Specify the appropriate search parameters to find the document that you added in the previous step, and click **Find Now**. See Figure 3-26 on page 102.

Archived

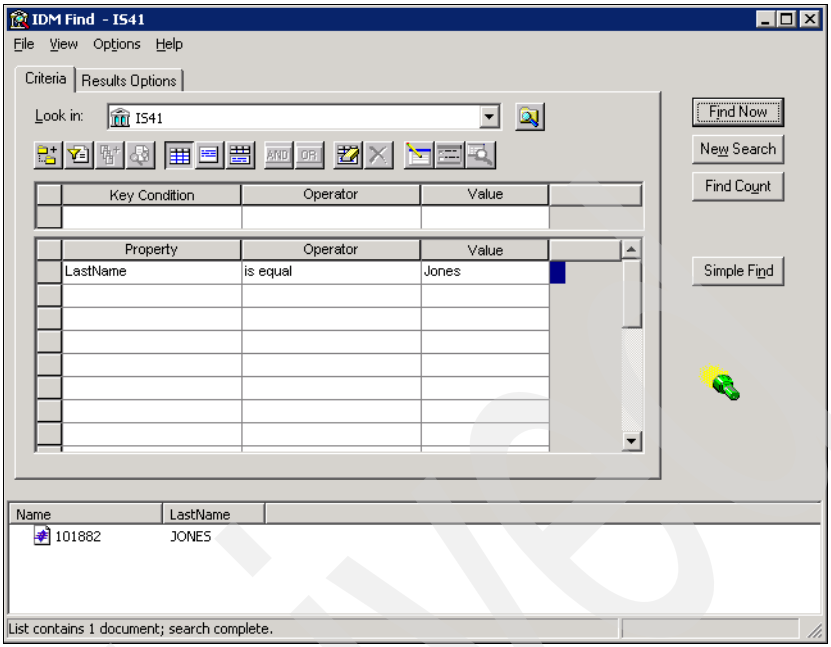



*Figure 3-26 Integrated Desktop Management Find*

5. Right-click the document in the results pane, and click **Properties**. Verify that the properties are correct.
6. Double-click the document in the results pane to view the document content.
7. Add an annotation, if desired.

## Search from WorkplaceXT

To verify that the document was federated, we can use multiple tools, including IBM FileNet Enterprise Manager, Workplace, and WorkplaceXT. We use WorkplaceXT in the following steps:

1. Log in to the Content Engine server or another system that can access WorkplaceXT with a Web browser.
2. Start your Web browser, and open the link to WorkplaceXT.
3. Log in with an appropriate user name.
4. Click **Search Mode** (magnifying glass icon) .
5. Select the configured object store from the drop-down list. In our example, the object store is **RbOS**.

6. We search for our document using the value in the mapped Last Name CFSIS property. Select **Advanced search** from the left pane. You can also use other search terms from the Simple search window.
7. Expand the **Search settings** list to expose the field condition window.
8. From the Match conditions drop-down list, select:

   **"Last Name CFSIS" "is equal" to "JONES"**

   > **Details:** We entered JONES in uppercase, because the Image Services index LastName was set to Convert to Upper. This Image Services index setting will convert all alpha characters to uppercase when adding the document. Integrated Desktop Management automatically converts searches to uppercase if the index is set to Convert to Upper, which is why we were able to search for the document by specifying `Jones`. The value was federated to Content Engine as JONES, so searches with WorkplaceXT must be performed in uppercase in this scenario. If the Image Services index is not set this way, searches must be performed using the same case that was entered when the document was added.

9. Click **Search**. See Figure 3-27 for the results.

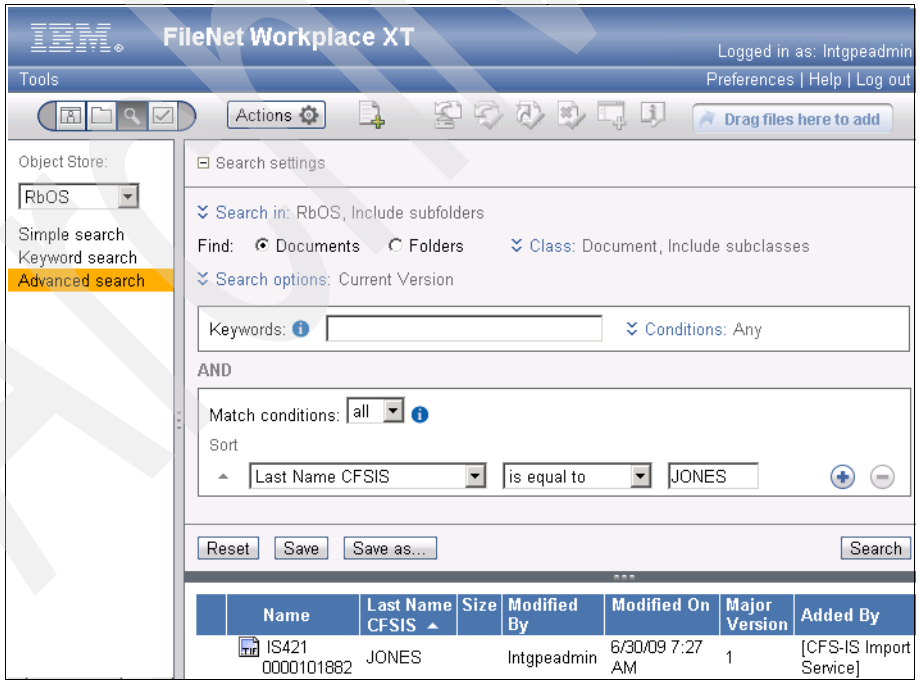



*Figure 3-27 WorkplaceXT advanced search for federated property*

10. Double-click the resulting document to view the content.
11. Notice that the Name of the document, which is also known as the Document Title, is IS421 0000101882, which is the FCD name followed by a space and the IS document ID with four leading zeros. This name provides another way to find the document by the IS document ID if you did not map the Image Services system index f_docnumber, but only if Document Title is not changed to use another value (which is possible).
12. Right-click the document, click **Properties**, and verify that the metadata matches the information that you entered when adding the document to Image Services.

## 3.5.5 Annotations

If your software versions are at least at Image Services 4.0 Service Pack 5 and Content Engine 4.0, annotations can automatically be federated. If you have federated documents using older software versions and then upgraded your software, you can federate the annotations separately using the Image Services export tool.

### Configure annotation export

Follow these steps to configure the annotation export process:

1. Log in to Image Services as an administrator user.
2. Start the Image Services Configuration Editor with either of these methods:
   - Select **Start** → **Run** → **fn_edit**.
   - Select **Start** → **Programs** → **FileNet Image Services** → **System Configuration** → **Configuration Editor**.
3. Click **OK** to accept the database and domain names.
4. Click the **System Appl. Services** tab.
5. Click the **Others** tab.
6. Click the red **X** in the Export Annotations field. The field will change to a green check mark. See Figure 3-28 on page 105.

> **Finding the Export Annotations field:** The Export Annotations field is on the far right side of the Others tab. Maximize the window, or use the scroll bar to see it.

Archived

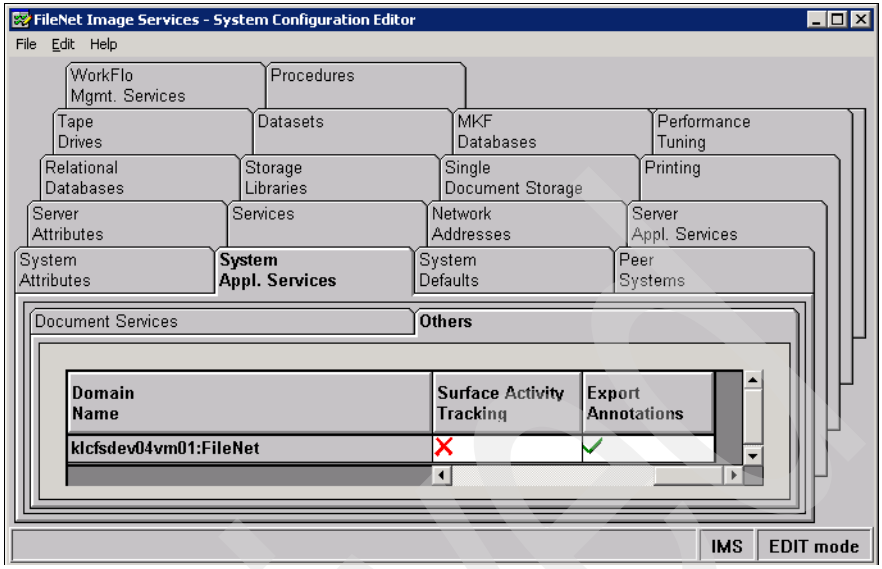



*Figure 3-28 Configure annotation export*

7. Click **File** → **Save**.
8. Click **File** → **Exit**.
9. Restart Image Services for the changes to take effect:

```
initfnsw -y restart
```

## Configure annotation security mapping

Without mapping annotation security from Image Services to Content Engine, all of the federated annotations will be set to allow full access to the Content Engine group #AUTHENTICATED-USERS.

If this approach is unacceptable, you can create the same security on the federated annotations that exists on the native annotations in Image Services.

### *Create security mapping*

In the following steps, we map the security of Image Services users and groups, which will be used for federated annotations:

1. Log in to Image Services as an administrator user.
2. From a command window, run the **SEC_map** command.
3. Export the Image Services users and groups:

   Run the **SEC_map> export_all** command:

– Enter the user name: Enter SysAdmin or another ID. This ID must have access to read the Image Services security database.

– Enter the password.

– Select the LDAP type: Enter `1` for Lightweight Directory Access Protocol (LDAP) or `2` for User Principal Name (UPN).

– Enter the prefix for the Distinguished Name.

– Enter the LDAP suffix.

– Enter the CE domain name.

Run the **`SEC_map> quit`** command.

> **Note:** To determine the prefix and suffix, start **FileNet Enterprise Manager**. Right-click the top level domain, and select **Properties**. Click the **Directory Configuration** tab, and then, click **Modify**. The first two characters of the value in the Directory Service User field are the prefix and everything after the user name is the suffix, except for the leading comma.
>
> For more information about the SEC_map tool, see the *IBM FileNet Image Services System Tools Reference Manual*, CG31-5612.

### *Verify and import the security mapping*

We view the created security mapping files, editing them if necessary, and import them into the Image Services security database. Follow these steps:

1. Use vi or Notepad to view and modify the two files that were created in the previous step (the `user.txt` and `group.txt` files). They were created in the directory from which the SEC_map tool was run.

   > **Note:** You can rerun the SEC_map tool and export the map again, if necessary. Delete or rename the existing `user.txt` and `group.txt` files before re-exporting the map.

2. From a command window, run **`SEC_map`**.
3. Import the map files:
   – Run the **`SEC_map> import`** command:
     - Enter the user name: Enter SysAdmin or another ID. This ID must have access to modify the Image Services security database.
     - Enter the password.
     - Enter the file name: `user.txt`

– Run the **`SEC_map> import`** command:

  Enter the file name: `group.txt`

– Run the **`SEC_map> quit`** command.

**Using the SEC_map tool:** The SEC_map tool validates each entry before it imports it into the security database. After an entry is imported, the corresponding entry is deleted from the `user.txt` file or the `group.txt` file. If an entry cannot be validated, it is not removed from the file. If all of the entries are valid, the SEC_map tool deletes the imported file. If any of the entries are invalid, those entries remain in the file. You can edit them to correct any problems and rerun the import.

### *Activate annotation security*

To activate annotation security, follow these steps:

1. Log in to Image Services as an administrator user.
2. Start Image Services **Application Executive** by typing `Xapex` at the command line.
3. Select **Applications** → **Security Administration**.
4. Select **System** → **Default Security Settings**.
5. Clear the selection from **CFS-IS No Annotation Security Mapping**.
6. Click **OK** to exit Security Administration.
7. Click **System** → **Exit**.
8. Click **File** → **Exit** to exit Application Executive.
9. Restart Image Services to make the changes take effect:

   ```
   initfnsw -y restart
   ```

## Configure federated annotation storage in P8

Unlike Image Services federated documents, the content of federated Image Services annotations is copied into the P8 object store. The P8 storage location into which this annotation content is copied is based on the storage policy set on the P8 ISAnnotation class. By default, this storage policy is the database storage policy. It can be changed in FileNet Enterprise Manager by navigating to ***<objectstore name>*** → **Other classes** → **Annotation** → **ISAnnotation** → **Properties** and modifying the storage policy for this class.

Archived

## 3.5.6 Closed documents

Closed Image Services documents are not included, by default, in the federation. All documents have a retention status of either active or closed and can be closed in one of two ways:

- When the document's retention is based on the date entered into the system and the retention period has been reached
- When the document's retention is based on the date that the document was closed and a user has manually closed the document using Integrated Desktop Management or another Image Services application

You can determine if a document is considered closed using Integrated Desktop Management.

Follow these steps to display the retention status of a document:

1. Start **IDM Find**.
2. Click the **Results Options** tab.
3. Select **Closed Documents**. By default, closed documents are not included in the search results. See Figure 3-29.

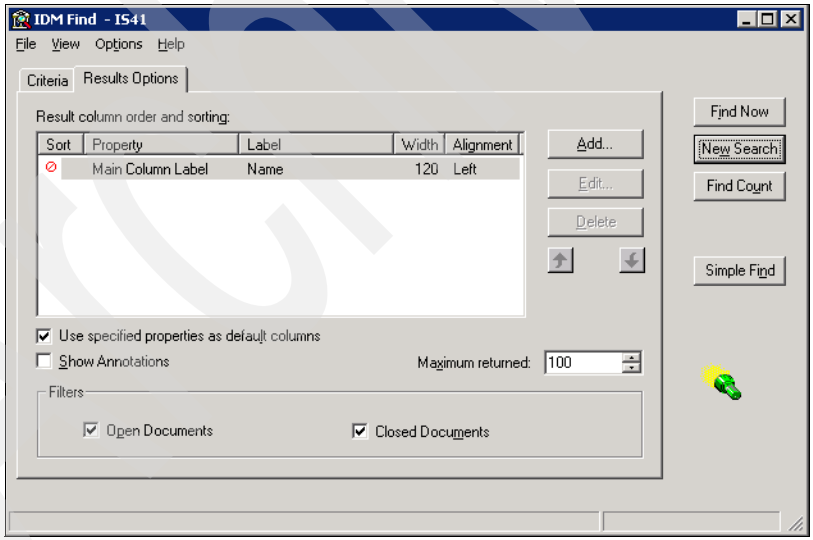



*Figure 3-29 IDM Find: Include closed documents*

4. Click the **Criteria** tab.
5. Specify the search parameters of your choice, and click **Find Now**.
6. Right-click one of the documents and click **Properties**.

7. The retention and expiration parameters are shown on the bottom of the window.

   For an example of an active document, see Figure 3-30.

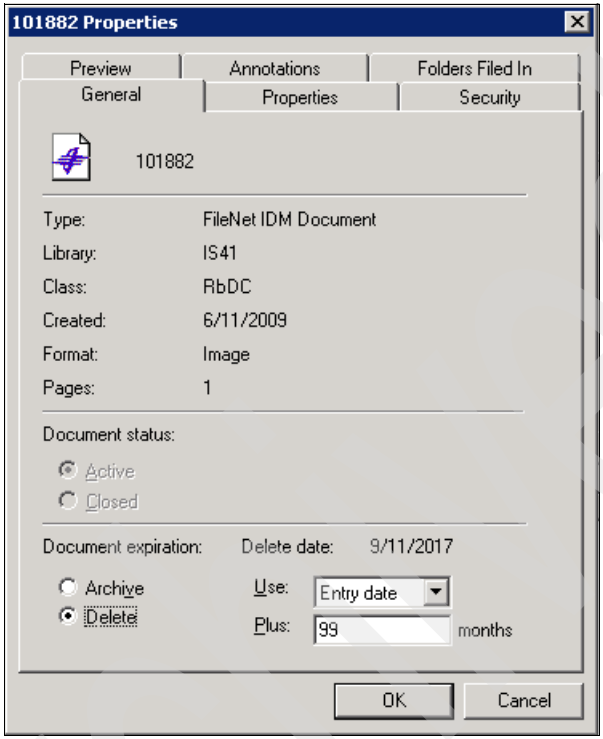



*Figure 3-30 IS document retention status: Active document*

For an example of a closed document, see Figure 3-31 on page 110. By default, this document will not be federated.

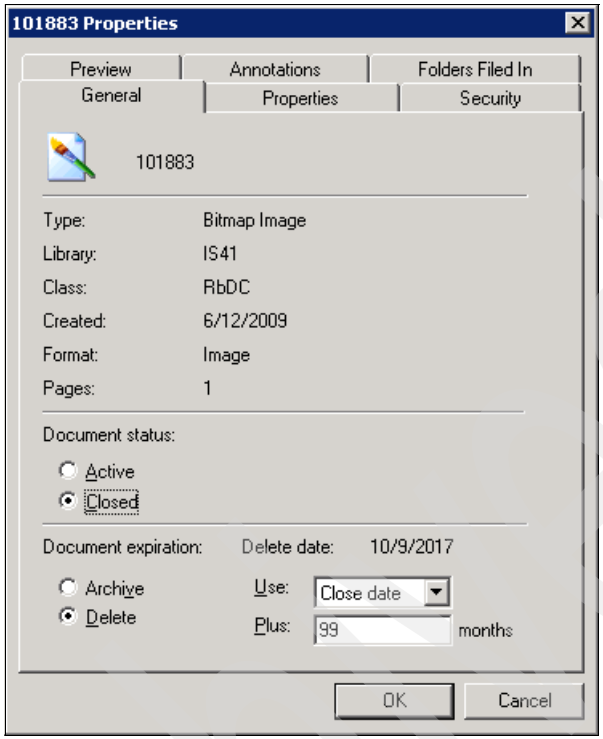



*Figure 3-31 IS document retention status: Closed document*

If you want to enable the federation of closed documents, follow these steps:

1. Log in to Image Services as the FileNet software administrator user.
2. Stop the Image Services software:

   ```
   initfnsw -y stop
   ```

3. Create an empty trigger file. The presence of this file will alert the Image Services catalog export process to export the metadata of both active and closed documents. The file does not need to contain any data:
   - On UNIX servers, use the **touch** command to create the trigger file with the name `/fnsw/local/trigger/INX_ce_export_all`:

     ```
     touch /fnsw/local/trigger/INX_ce_export_all
     ```

   - On Windows servers, use Notepad to create the trigger file with the name `\fnsw_loc\trigger\INX_ce_export_all`. See Figure 3-32 on page 111.

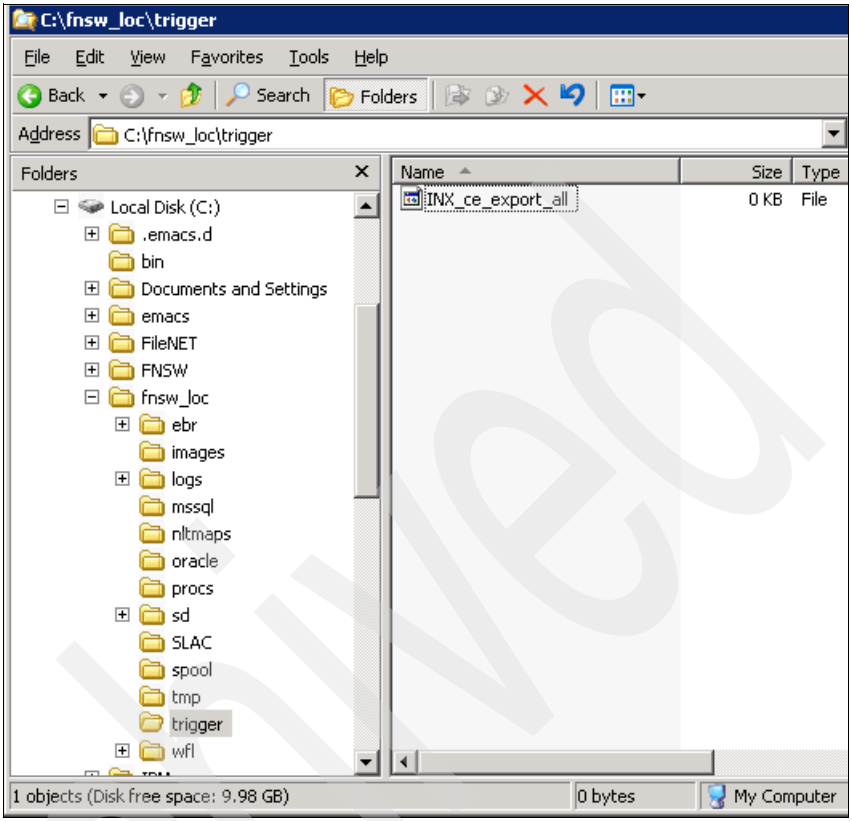



*Figure 3-32 INX_ce_export_all trigger file*

4. Start the Image Services software:

   `initfnsw start`

5. View the Image Services event log (elog) to verify that the `INX_ce_export_all` trigger file has been enabled. See Example 3-1 for an example from the elog from our system.

   *Example 3-1 IS elog acknowledgment of trigger file*

```
2009/07/09 09:32:54.171 90,0,111 <fnsw> INXu  (3692.7668.60
0xe6c.1df4) ... [INFO]
CE INX_export CLOSED and ACTIVE documents. Trigger file
<c:\fnsw_loc\trigger\INX_ce_export_all> found
```

6. Run the Image Services export tool as described in the previous section. If you have already federated documents in a document class that contains closed documents, run the catalog export tool again for that document class,

Archived

but do not select the Re-Export option. The catalog export tool will skip the active documents that have already been federated and will federate only the closed documents.

## 3.6 Federated records management with Content Federation Services for Image Services

CFS-IS provides the bridge that is required to records enable federated Image Services content. Using Enterprise Records (formerly known as IBM FileNet Records Manager) in conjunction with CFS-IS, you can declare, secure, and manage federated records from Image Services. This section provides information about configuration, lockdown, and the deletion of records using CFS-IS.

### 3.6.1 Lockdown behavior

*Lockdown* is the action of protecting a record and its associated document from any changes or deletion before its disposition. It is a critical component of a records management system.

If you use Enterprise Records in conjunction with CFS-IS and an Image Services system, you can lock down any CFS-IS documents that have been declared as records. Therefore, these documents can no longer be modified or deleted during the records retention period. Users cannot use the Image Services database maintenance tools to update or delete the documents, regardless of the documents' assigned date. Lockdown of an Image Services (IS) document occurs automatically as a background process. When a document is declared as a record by Enterprise Records, the Image Services document's security attributes are altered to prevent unauthorized modification or destruction of the content, metadata, and annotations, until the content has reached its end of life. The lockdown of annotations is optional. It is possible to lock down the content and metadata but to allow for the creation of additional annotations on locked content, depending on your business requirement.

During the configuration of CFS-IS, we recommend that you create a records administrator user and group. Members of the records administrator group are the only users with the ability to delete content in Image Services. This capability is provided for cases where documents are incorrectly declared as records and need to be deleted or for testing purposes. Access to this group is often restricted to the CFS system account only. CFS uses this account to lock down the content and destroy the content when it reaches the end of life. Securing the Image Services content is therefore achieved by replacing the write and

append/execute group IDs with the newly created records administrator group. Anyone who is not a member of the records administrator groups has controlled access to the declared documents.

For example, you can create a user called CFSRM, a group called RecordAdministrators, and add the user CFSRM to the RecordAdministrators group. When documents are declared records, which means the records are locked, the documents become read-only to predefined groups. You can replace groups with Write or Append/Execute rights with the RecordAdministrator group, so that users in that group become records administrators for the documents. Figure 3-33 shows an example of the security attributes of a document before and after record declaration. The names CFSRM and RecordAdministrators are only used as examples in this context. You can configure your system according to your business requirement.

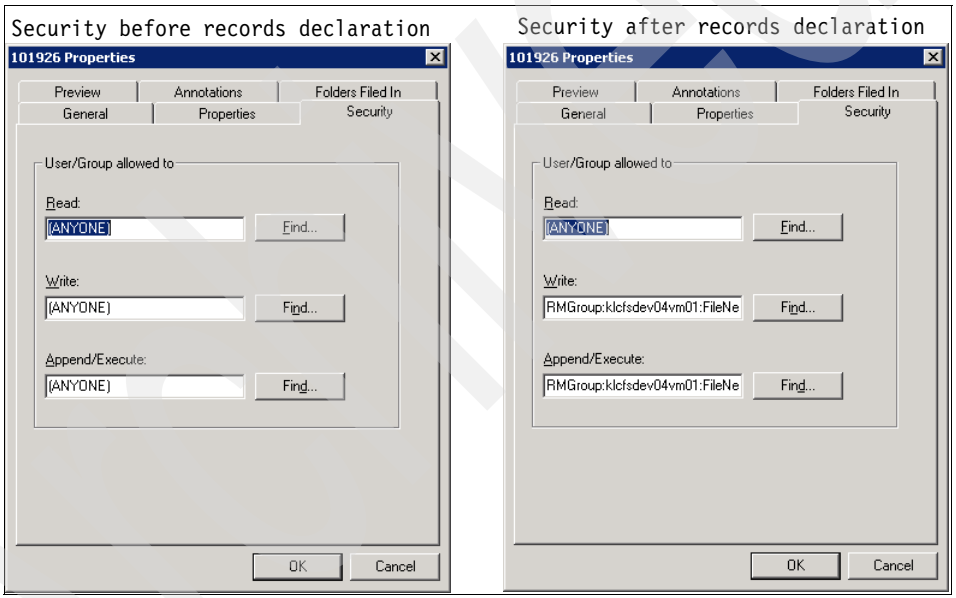



*Figure 3-33 Security before and after record declaration*

**Performance consideration:** Always consider the extra system load that is added to the existing system when working with records. For instance, the lockdown operation, when performed on a large number of federated documents, can consume significant system resources, which need to considered. Frequent record searches by records administrators can also be costly. The Enterprise Records sweep processes (disposition and hold) can also greatly increase the system load. Plan ahead to insure that you have sufficient hardware to handle your operations and that you have sufficient time scheduled to complete these operations.

## 3.6.2 Deletion behavior

At disposition time, Enterprise Records verifies that the document is eligible for destruction. To be eligible, the retention period has expired, and there must not be any holds on the record.

When a record reaches its disposition phase and its destruction is approved, Enterprise Records instructs CFS to dispose of the content. The system then waits for confirmation from the Image Services server before it deletes the record metadata on the record object. The assumption is that Image Services expunges the content from magnetic or optical media when and where allowed so that forensic discovery tools at the operating system level cannot recover the record content.

## 3.6.3 Image Services configuration

To configure federated records management for CFS-IS, you adjust security in Image Services and test the functionality using IBM FileNet Content Manager and Image Services. This section covers Image Services-related configuration, which consists of creating a new group and then assigning Enterprise Records security to it.

### Creating a group for Enterprise Records

Create a new group in the Image Services security database for use by Enterprise Records with the following steps:

1. Log in to Image Services as an administrator user.
2. Start Image Services Application Executive by typing `Xapex` at the command line.
3. Select **Applications** → **Security Administration**.

4. Select **Groups** → **Add Group**, and assign a name. In our example, we chose `RMGroup`.
5. Next to Group's Members, click **Add,** enter the name of the CFS-IS administrator user, and click **OK**. In our example, we used `CFSAdmin`. See Figure 3-34.

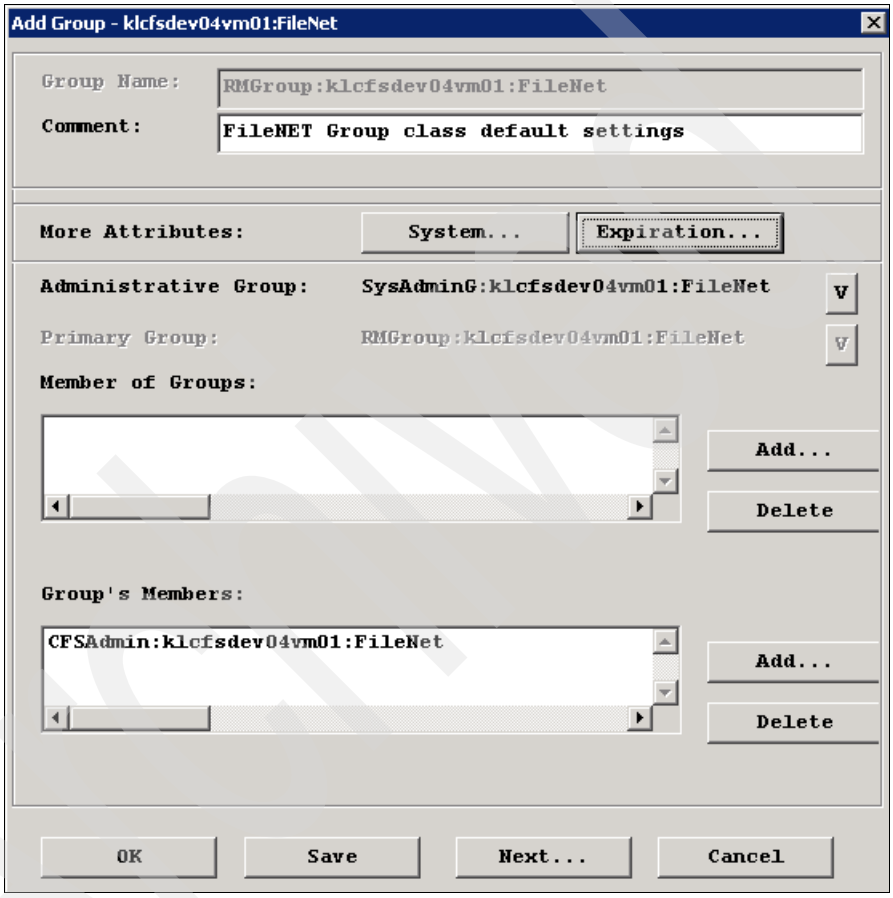



*Figure 3-34 IS security group for use by Enterprise Records*

6. Click **OK** to create the group. You get a message that the group was added successfully.
7. Click **System** → **Exit**.

## Configuring Image Services security for Enterprise Records

Follow these steps to configure Image Services to use the Enterprise Records group (that you created in the earlier step) when documents are locked by Enterprise Records.

> **Important:** After you establish Enterprise Records security, you cannot remove it. Be cautious when you configure the security.

To configure Image Services to use the Enterprise Records group:

1. Log in to the Image Services server as an administrator user.
2. Open a command window.
3. Type the following command:

   `SEC_rm_config`

4. Type `e` to edit the configuration settings.
5. Select the desired record-level security. We entered `s` for READ_ONLY.

   > **Record-level security:** The default setting for Record Level Security is READ_ONLY, which is the highest level of security. There are three levels of security:
   >
   > - **READ_ONLY**: Any user that is not a member of the SysAdminG group and who already has at least Read privileges to the documents can only view the documents. This level is the default. Type `s`.
   > - **APPEND_ONLY** (includes Read permission): Any user that is not a member of the SysAdminG group and who already has at least Read and Append/Execute privileges to the documents can only view the documents and add new annotations. Type `w`.
   > - **NO_CHANGE**: No additional security is applied. The same security settings that currently exist in the Image Services index database apply. Type `n`.
   >
   > To set security, type the letter of one of these security levels, or press Enter to accept the READ_ONLY setting.

6. For the FileNet Record Manager group, type the group name that you created in the previous step. In our example, we entered `RMGroup`.
7. Choose either MINIMAL or VERBOSE activity logging. We pressed Enter to accept the default, which is MINIMAL.
8. Optionally, type `d` to display the current settings before saving.

9. Type s to save the new settings.
10. Type q to quit or exit the utility.

Example 3-2 shows our results.

*Example 3-2 SEC_rm_config utility example*

```
C:\>SEC_rm_config

Image Services Record Management Configuration Utility

Command line options:

 d     Display current configuration settings
 e     Edit configuration settings
 s     Save configuration settings
 p     Print Menu

 q     Quit


Enter command => e


Edit Record Level Security

s   READ_ONLY
w   APPEND_ONLY
n   NO_CHANGE
Enter Record Level Security [READ_ONLY] => s


Edit Record Manager Group

Enter Record Manager Group Name [UNDEFINED] => RMGroup



Edit Activity Log Level

m   MINIMAL
v   VERBOSE
Enter Activity Log Level [MINIMAL] =>
```

Archived

```
Enter command => s


Saving Record Management control table information...
SEC_rm_config: Record Management settings have been successfully
updated


Enter command => d


Current Image Services Configuration:
Record Security Level     : READ_ONLY
RM Group                  : RMGroup
RM Log Level              : MINIMAL

Current User Selected Configuration:
Record Security Level     : READ_ONLY
RM Group                  : RMGroup
RM Log Level              : MINIMAL


Enter command => q

Exiting...
C:\Documents and Settings\Administrator>
```

**Important:** When using a CFS-IS system with Enterprise Records, do not set an expiration date on documents that are stored in Image Services. There is currently no automated way for the CFS-IS administrator user on the Image Services system to differentiate between record documents that are placed on hold that cannot be deleted and those record documents that can be deleted.

### 3.6.4 Content Engine configuration

You do not have to make any additional configuration changes in IBM FileNet Content Engine to enable Enterprise Records functionality with federated documents. The installation and configuration of Enterprise Records is outside the scope of this document.

Archived

For help with installation, configuration, and the use of Enterprise Records, see the following resources:

- Product documentation for FileNet P8 Platform at this Web site:

  http://www.ibm.com/support/docview.wss?rs=3273&uid=swg27010422
- Click **ecm_help** → **Expansion Products** → **IBM Enterprise Records**.
- *Understanding IBM FileNet Records Manager*, SG24-7623

## 3.6.5 Testing lockdown

To test the effect of lockdown (declaring a federated document as a record) on a federated document's Image Services security, we viewed the Image Services security before and after lockdown. Follow these steps to test your system:

1. Use Integrated Desktop Management to find a federated document that has not been locked (declared as a record).
2. View the security of the document. See Figure 3-35.

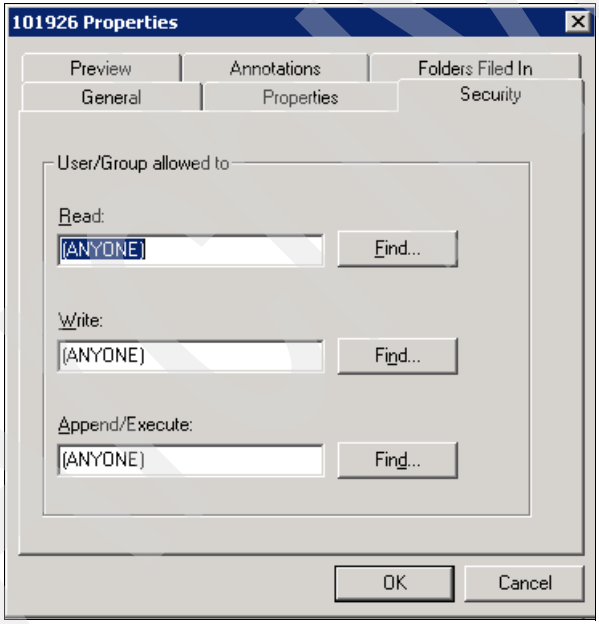



*Figure 3-35 IS security of federated document before lockdown*

3. Use Enterprise Records to declare the document as a record. We used the Enterprise Records functionality within WorkplaceXT. See Figure 3-36 on page 120.

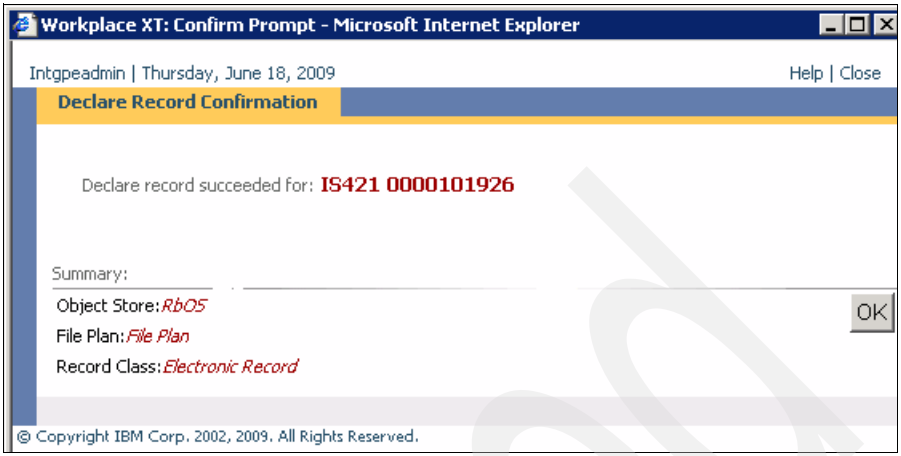



*Figure 3-36 Declaring the document as a record*

4. Use Integrated Desktop Management to find the same document in Image Services.

> **Tip:** Do not select the properties again from the same results window that was used in the previous step, because Integrated Desktop Management caches the properties. You must issue a new search.

5. View the security of the document. Notice that the security for the Write and Append/Execute fields is now set to the IBM Enterprise Records (formerly known as IBM FileNet Records Manager) group that is specified by the SEC_rm_config process, which, in our case, is RMGroup. See Figure 3-37 on page 121.

Archived

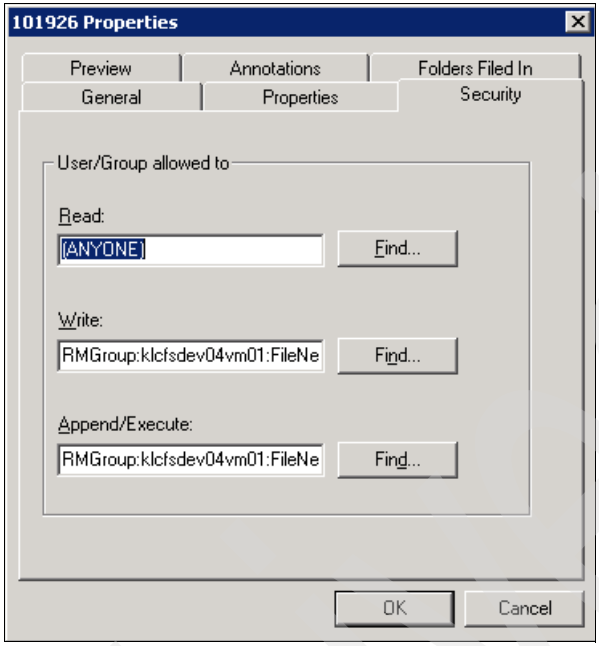



*Figure 3-37 IS security of federated document after lockdown*

## 3.7 Troubleshooting

You might run into problems when using CFS-IS. This section describes various ways to troubleshoot the problems.

### 3.7.1 Log files

When you try to determine why a document did not federate as you expected, you can review various log files for errors or messages that might indicate the reason. There are relevant log files in both Image Services and Content Engine:

- Image Services event log (elog)
- Content Engine error log
- Content Engine trace log

#### Image Services event log (elog)

IS writes error messages, including those error messages that are related to CFS-IS, to a log file named `el`*yyyymmdd*, where *yyyy* is the four-character year, *mm* is the month, and *dd* is the day.

The file is located in one of these directories:

| | |
|---|---|
| **UNIX** | `/fnsw/local/logs/elogs` |
| **Windows** | `\fnsw_loc\logs\elogs` |

> **Tip:** At a command prompt on the Image Services system, type `vl`, and the current elog is shown.

## Content Engine error log

The Content Engine writes error messages, including those error messages that are related to CFS-IS, to a log file named `p8_server_error.log`. The file is located, by default, in the application server instance's working directory path appended with `/FileNet`. On our system, the path was `C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\FileNet\server1`.

## Content Engine trace log

Typically, you can find the error based on the messages from the Image Services or Content Engine error logs, but if you want to see more information about the federation process, you can enable trace logging in Content Engine. If tracing is enabled for the subsystems that are used by CFS-IS, Content Engine writes detailed information about work that is processed during federation. You can configure tracing for various levels of detail and for multiple individual subsystems. Disable tracing except when you troubleshoot a problem, because tracing can have affect system performance negatively, and the trace logs grow quickly.

To enable trace logging, follow these steps:

1. Launch FileNet Enterprise Manager Administration Tool by selecting **Start** → **Programs** → **IBM FileNet P8 Platform** → **FileNet Enterprise Manager Administration Tool**.
2. Right-click the top level P8 domain, and click **Properties**.
3. Click the **Trace Control** tab.
4. Click **Enable Trace Logging**.
5. In the Log File Output Location field, enter an existing path or leave the field unchanged. If you leave the field unchanged, the file will be written to the path that was last specified. If a path was never explicitly specified, the file will be written to the same location as the Content Engine error log default location.

Archived

6. Select the subsystem for which you want trace logging. For CFS-IS, select **CFS Daemon**. If your problem is caused by another component of Content Engine (database, content-based retrieval, security, and so forth), you might need to enable the tracing of other subsystems, as described in this document:

   http://www-01.ibm.com/support/docview.wss?rs=3278&uid=swg21313968

   See Figure 3-38.

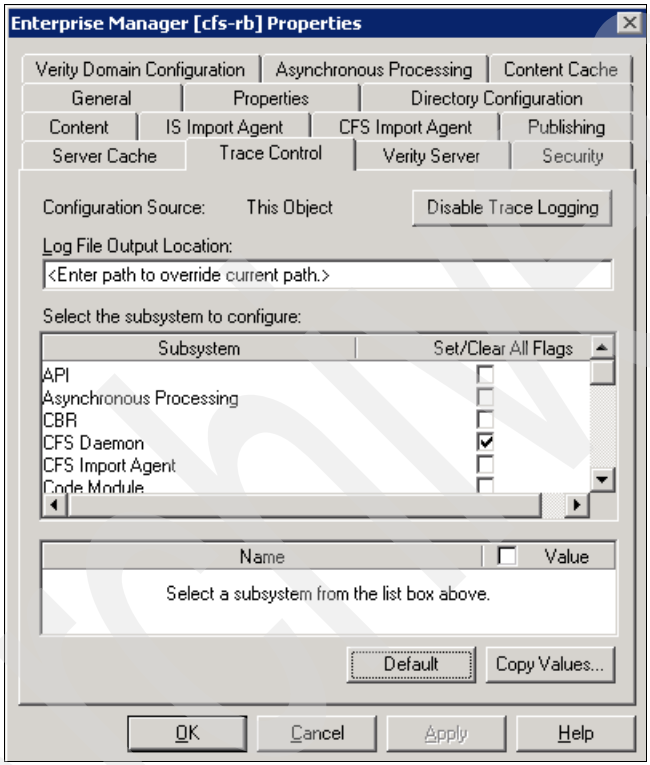



*Figure 3-38 Content Engine trace logging control*

## 3.7.2 Image Services database table export_log

When you add a document or modify its metadata, the Image Services export process will add an entry in the CFS export queue, which is a table in the index database named export_log. If certain errors occur or if the Image Services Import Agent is disabled in Content Engine, the document reference will be added to and remain in the export_log table. The export_log table is a good place to look when you troubleshoot problems.

In the following steps, we verify that our test document is not in the table:

1. Log in to the Image Services server or another system that can access the Image Services database tables. Our Image Services system used Microsoft® SQL Server 2000, so we used SQL Enterprise Manager to view the contents of the table. Use the appropriate SQL tool for the database that your Image Services system uses.

   When referencing the export_log from SQL tools, you need to specify the owner, for example:

   ```
   select * from f_sw.export_log;
   ```

2. Start SQL Enterprise Manager or another database tool.
3. Drill down to the Image Services index database (often named indexdb) and the export_log table.
4. Open the table to verify that there are no entries. See Figure 3-39.

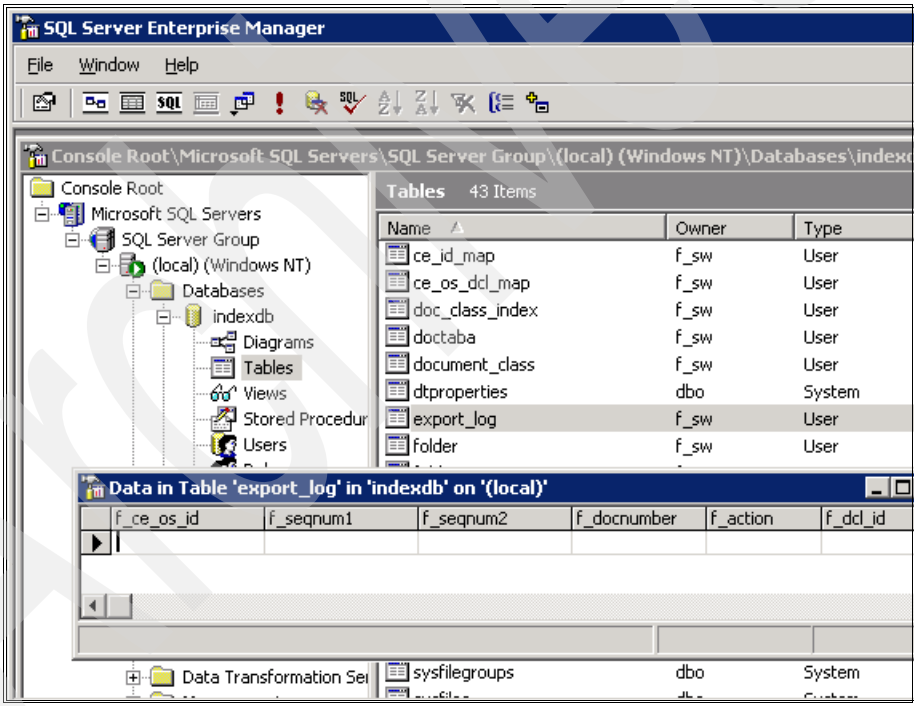



*Figure 3-39 IS export_log table*

If you see entries in the table, it might be helpful to know what the fields indicate. Table 3-6 on page 125 describes the table.

*Table 3-6 Export_log table*

| Column name | Description |
|---|---|
| f_ce_os_id | Internal Image Services ID that is used to track each object store instance |
| f_seqnum1 | Sequence number 1 |
| f_seqnum2 | Sequence number 2 |
| f_docnumber | IS document number |
| f_action | Action code:<br>1 = Insert<br>2 = Export<br>3 = Update<br>4 = Export and delete from doctaba table<br>5 = Delete from Content Engine |
| f_dcl_id | Document class ID value that is used to track the Image Services document class |
| f_cat_in_doctaba | Signifies whether cataloged in the doctaba table |
| f_new_dir | New updated index values |
| f_old_dir | Old index values |

### 3.7.3 Image Services database table doctaba

If you want to know whether, or where, a document has been federated, the Image Services table that contains document IDs (doctaba) contains a column (f_ce_os_id) containing the object store ID of the most recently federated object store.

Archived

### 3.7.4 Content Federation Services status

Use these methods to determine if CFS-IS is running:

- CFS-IS 3.5.*x*

  On the Content Engine, start the CFS Connection Monitor.
- CFS-IS 4.*x*:
  - Performance counter data: CFS-IS reports performance and uptime data that can be viewed by IBM FileNet System Monitor or IBM System Dashboard for Enterprise Content Management.
  - Enable tracing on Content Engine

### 3.7.5 Common errors

**Note:** With CFS 4.*x*, when the Content Engine detects invalid data, the system typically reports the error in the error log, removes the entry from the export_log table, and continues the federation process. In certain cases, however, the entry will remain in the export_log table.

With CFS 3.5.*x*, when the Content Engine detects invalid data to be federated, the system terminates the Import Agents and leaves the items in the Image Services export_log table. The user is expected to resolve the error before resuming the federation process. If you want the agents to continue to process work after encountering errors, you must set a registry parameter to set the number of errors that is allowed before terminating:

Select **HKEY_Local_Machine** → **SOFTWARE** → **FileNet** → **ECM** → **Content Engine** → **Image Services** → **CFS-IS Error Threshold**.

Set this parameter to a value between `1` and `2147483486`.

The following list shows the common types of errors that are encountered during federation.

## Data mismatches

The following list shows common data mismatch errors:

- **Field size mismatches**

  If the data for an Image Services index is larger than the maximum allowed by the property in Content Engine, you will get an error similar to this error in the `p8_server_error.log` file:

  ```
  Caused by: com.filenet.api.exception.EngineRuntimeException:
  E_BAD_VALUE: The value given for a property or list element lies
  outside the permitted range or value set, or exceeds the maximum
  length allowed. The length of the value (26) specified for property
  LastName exceeds the maximum permitted length (10).
  ```

  If this situation occurs, increase the maximum allowable value for the property, and re-export the document.

- **Menu item mismatches**

  If the data for an Image Services menu index does not have a corresponding choice list item in Content Engine, you will get an error similar to this error in the `p8_server_error.log` file:

  ```
  Caused by: com.filenet.api.exception.EngineRuntimeException:
  E_BAD_VALUE: The value given for a property or list element lies
  outside the permitted range or value set, or exceeds the maximum
  length allowed. The value (NT) specified for property
  Rb_dc_CFSIS.State_CFSIS is not within the range of permitted values.
  ```

  If this situation occurs, add the choice list item to the choice list, and re-export the document.

- **Date mismatches**

  If the date in an Image Services date index contains a value that is older than is acceptable in Content Engine (December 31, 1752), you will get an error similar to this error in the `p8_server_error.log` file:

  ```
  Caused by: com.filenet.api.exception.EngineRuntimeException:
  E_BAD_VALUE: The value given for a property or list element lies
  outside the permitted range or value set, or exceeds the maximum
  length allowed. The value (Fri Jul 01 04:00:00 PST 1009) specified
  for property PolicyDateCFSIS is less than the minimum permitted (Sun
  Dec 31 16:00:00 PST 1752).
  ```

  If this situation occurs, replace the dates in Image Services with an allowable value, and re-export the document.

Archived

In these data examples, you will get a corresponding error in the Image Services elogs similar to this error:

```
2009/06/18 09:57:59.191 90,6,31 <fnsw> INXs  (6128.4148.101
0x17f0.1034) ... [SERIOUS]

INXl_ce_get_next_logs(line=802) Document 103220 could not be migrated
to CE by the IS_import_agent. For more information on the problem refer
to the error log on the Content Engine Server. Once the problem is
resolved, the document will need to be re-exported. The export_log
action=3 Host_id='server1:CommitDocs_RbOS_#18' CE_os_id=1000 Object
Store name='RbOS' GUID={CC3BA159-3460-4AEB-A78F-3AEBA0A13815} CE
Domain_name='cfs-rb'  GUID={233A1549-1721-4B6E-A411-93AE36E592EC}
```

## Other errors

This list describes other errors that might occur during federation:

- **Large volume of annotations fill up the MKF Permanent DB recovery log (PermDB)**

  When federating annotations, the content is included, which differs from documents, in which only metadata, and not content, is federated. During the export process, the annotations are queued in a table named annot_log in the PermDB. During a large export of annotations, you can fill up the PermDB recovery log. If so, you will see an error in the Image Services elog similar to this error:

  ```
  2009/06/08 03:27:47.803 161,0,1327 <fnsw> DOCs  (5627) ...

  WARNING:  CAN NO LONGER ROLL DATABASE /fnsw/dev/1/permanent_db0
  FORWARD!!

  Some recovery log data has been overwritten, thereby introducing a
  gap. Make the recovery log larger, unless you have skipped more than
  two backups.  If you do not, then, when you restore a backup, you
  will lose all processing since the backup was made!!  The recovery
  log (that is, aij) must be large enough to hold all recovery log
  information generated for two backup periods.  Two backup periods is
  normally two days:  At most sites, interval backups must be done at
  least once a day, and full backups must be done once a week.
  ```

  To resolve this situation and to avoid this problem in the future, increase the size of the PermDB recovery log.

- **DB_NOT_UNIQUE errors**

  If you encounter an error similar to the following error in the `p8_server_error.log` file, it might occur because you re-exported documents that have been successfully federated in the past:

```
2009-06-18T17:09:01.343Z 2EF72EF7 CFSD FNRCE0236I - INFO CommitDocs
updateBatch exception
com.filenet.api.exception.EngineRuntimeException: DB_NOT_UNIQUE: The
update or insert failed due to an attempt to create a duplicate value
in a unique index. Statement List:

  SQL: "INSERT INTO IndexRequests (target_id, target_id_class,
operation, indexation_id, status, retry_count,
element_sequence_number, index_area_id, request_time,
seqnum_identity) VALUES (?, ?, ?, ?, ?, ?, ?, ?, CURRENT TIMESTAMP,
NEXTVAL FOR CISequence)" In-bindings:
({00019334-F01A-5121-A0D2-4E240F1FAB21},{1FEE687A-ACD2-4E06-B138-DFF
325A3F6DB},0,{B5AA0637-B736-4EBC-9053-282B65714674},0,0,0,{2BCCED88-
375A-4027-BB1E-F47B057240DE})

  SQL: "INSERT INTO DocVersion (object_class_id, epoch_id, creator,
create_date, modify_user, modify_date, object_id, is_reserved,
is_current, is_frozen, versioning_enabled, major_version_number,
minor_version_number, version_status, checked_in_date,
storage_class, mime_type, is_in_exception_state,
classification_state, storage_area_id, indexation_id,
compound_document_state, u34_documenttitle, u62_candeclare,
ua3_isdocnumbercfsis, ub1_lastname2, ub2_state2_cfsis, security_id,
content_info, component_types, retrieval_names,
content_referral_blob, next_content_unique_id, version_series_id,
dynamic_cr_update_status, content_info_2, component_types_2,
retrieval_names_2) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, null,
null, null)" In-bindings:
({1FEE687A-ACD2-4E06-B138-DFF325A3F6DB},0,'[CFS-IS Import
Service]',Thu Jun 18 10:09:00 PDT 2009,'[CFS-IS Import Service]',Thu
Jun 18 10:09:01 PDT
2009,{00019334-F01A-5121-A0D2-4E240F1FAB21},false,true,false,true,1,
0,1,Thu Jun 18 10:09:01 PDT
2009,{CAB6558A-98B7-4C2A-8CB2-2A784B4759CE},'text/plain',false,0,{D8
A76A1F-59E1-43A2-A8F0-60CA5E8B846C},{B5AA0637-B736-4EBC-9053-282B657
14674},0,'IS421
0000103220',true,103220.0,'smith','GA',{17A8E421-324A-4B1B-BA71-DE2A
7B5520ED}, 0x00000000feffffffffffffff,
0x140001007400650078007400 2f0070006c00610069006e000000,[len=50]
0x2c000100490053003400320031002000300030003000300031003000330032 00..
.,[len=44]
0x010002001c060000020000000349301001af02151a0d24e24f1fab2134930100...
,1,{A20D7A1E-7A65-466D-8686-AFB8B1A6FF81},0
```

This error is a normal error and will not cause any problem. When documents are federated (export or re-export), there is no check to see if the document already exists. This is for performance reasons. If the document exists, Content Engine reports this error, and overwrites the Content Engine properties of the existing document, if it exists.

## 3.8 Tuning

In this section, we describe various concepts that are related to tuning, bulk import, remote caching, and capacity planning.

### 3.8.1 Bulk import

When importing a large number of documents from Image Services, you must consider several factors that can significantly affect the amount of time that it takes to complete the import.

#### IS catalog export

The first step in the federation process is to specify to Image Services which documents it must export. This process is called the *catalog export*. This process can be a slow and resource-intensive process. The export process works primarily with the doctaba table, which contains the metadata for all of the documents in the Image Services system and is typically an extremely large table. When you use the export tool to specify a set of documents to federate, the export process must analyze all of the documents for a particular document class to determine whether they must be federated. Because of the nature of the query, there are no viable retrieval keys that the database can use to prevent a full table scan. Full table scans can be extremely time-consuming.

To improve the performance of this process, you can create a database index (retrieval key) on the doctaba table, which will help the export process complete its processing more quickly, because using an index prevents a full table scan. However, like most actions involving performance improvements, creating an index can cause performance degradations elsewhere. Adding an index requires additional database space to hold the index data and reduces the performance of the inserts into the table and the deletions from the table.

The following steps describe how to create this index:

- **Oracle**

```
create index f_docn_doccn_ix on f_sw.doctaba
("F_DOCCLASSNUMBER","F_DOCNUMBER") tablespace <table_space>;
```

- **SQL Server**

  ```
  create index f_docn_doccn_ix on f_sw.doctaba
  ("F_DOCCLASSNUMBER","F_DOCNUMBER") on <file_group>
  ```

- **DB2**

  ```
  create index f_docn_doccn_ix on f_sw.doctaba
  ("F_DOCCLASSNUMBER","F_DOCNUMBER")
  ```

We recommend dropping these indexes after all of the existing documents have been federated.

## Content Engine Import Agent

Content Engine provides a number of variables for tuning the CFS-IS Import Agents. The default settings are likely inadequate when importing a large number of existing Image Services documents. There are three settings that can have the most impact in performance:

- **Number of Import Agents**

  Increasing the number of Import Agents increases the overall throughput of the import. However, increasing the number of Import Agents also has these effects:

  – Increasing the number of Import Agents increases the number of connections to the Image Services server.

    Each Import Agent has two connections: one connection for documents and the other connection for annotations. For example, four Import Agents on a Content Engine server use eight Image Services connections.

  – Increasing the number of Import Agents increases the Java virtual machine (JVM) memory that is consumed on the application server.

    You can monitor the application server heap size usage and increase the maximum heap size, if necessary and possible.

  – Increasing the number of Import Agents increases the contention for each federation.

    Because of database, memory, and CPU contention to create each document in the object store, it might take slightly more time to federate each document.

- **Documents per batch**

  The batch size specifies the number of documents that are federated in a batch. Increasing this parameter causes each Import Agent to federate a larger number of Image Services documents at a time, which requires fewer round trips between the Image Services and Content Engine servers. However, making the batch size too large increases the chance that an Import

Archived

Agent will not have enough time to federate an entire batch within the Image Services “Doc services” timeout period. If a batch is not fully federated within the timeout period, it can be reprocessed by another Image Services Import Agent. A second Image Services Import Agent reprocessing a batch that is not fully federated results in duplication errors, which are automatically rolled back. It also needlessly increases the CPU and database contention, because of the duplicate commit actions and subsequent rollback actions.

- **Batch delay**

  The batch delay parameter specifies the number of milliseconds that the Import Agent waits between batch requests before retrieving the next batch from Image Services. A smaller batch delay results in more requests in a given amount of time and can be an alternative approach to adding more Import Agents.

To help you understand how these factors work together, we provide this explanation of the process. The Import Agents either process documents at the batch level or at a single document level. The fewer database transactions required to complete the batch, the higher the throughput will be. The best throughput occurs when the agent can process the entire batch with a single database transaction, which usually occurs when importing existing and new Image Services documents that do not have any metadata updates.

There are five primary action codes associated with the Import Agents:

- Insert (1)
- Export Only (2)
- Update (3)
- Export and Delete (4)
- Delete (5)

The Content Engine processes a batch of documents consisting of only inserts as a single insert transaction to the database for the entire batch. The Content Engine also processes a batch of documents consisting of inserts and exports as a single transaction to the database for the entire batch. However, in all cases, if any error is returned, the Import Agent will process the entire batch in single document mode. Update and delete actions are processed at the document level. Therefore, a batch of documents with updates and deletes will require many more transactions and, therefore, more time to complete.

For example, consider a batch that has the following action codes (1, 1, 1, 3, 3, 1, 1, 1, 1). Assuming that no errors are encountered, the batch requires six transactions:

- Transaction 1: Insert first three documents
- Transaction 2: Retrieve document from database
- Transaction 3: Update document

- Transaction 4: Retrieve document from database
- Transaction 5: Update document
- Transaction 6: Insert last four documents

To get the best performance throughput, try various combinations of the parameters. Setting each parameter to its “highest performance” setting does not always result in the best throughput.

When there are many updates in the batch, a small batch size results in the best throughput. When you process all new documents, a larger batch size is best.

In a farmed environment with numerous Import Agents, we have observed that performance improves when the batch size is adjusted in the opposite direction of the number of Import Agents, especially when the total agent count for any single object store exceeds 25. *Therefore, if you increase agents, also decrease batch size. If you increase batch size, decrease the number of agents.*

When the Import Agent receives a batch of documents that have action codes consisting only of inserts and exports, the Import Agent attempts to insert the entire batch into the Content Engine database in a single transaction. If there are no “repeated” inserts, and all of the exports, if any exist, are not re-exports, the insert is successful, which is the scenario that results in the best throughput.

However, if an error is returned, the entire batch will be processed at the single document level. For every document in the batch, the Import Agent performs a retrieve from the database, examines the data, and then updates the database. Therefore, in this scenario, the number of transactions for this batch is $2N + 1$, and $N$ is the number of documents in the batch. This situation is the worst case scenario for a batch that had not encountered a timeout condition.

We recommend that you avoid timeout conditions if at all possible. If you encounter timeout messages from the Import Agent, try reducing the batch size.

Again, you might need to try several combinations of parameters to achieve the best possible performance.

The parameters are located in the Image Services Import Agent tab of the P8 domain in IBM FileNet Enterprise Manager. See 3.5, “Configuration” on page 69 for details.

## Property indexes

Content Engine property indexes (retrieval keys) will significantly improve query performance when searching for documents, but they can affect the performance of a bulk import of documents from Image Services. Wait until after you have federated the bulk of your Image Services documents before you create indexes on Content Engine properties.

### Business process management

You can initiate workflow from federated documents, but we recommend that you do not enable this function during a bulk import of documents, because it might affect the speed of the federation. Wait until after the bulk import is complete before enabling workflow on federated documents.

## 3.8.2 Remote caching

If your Image Services server is in a separate physical location from the Content Engine, you can improve the performance of document retrievals by creating an Image Services cache on the Content Engine server or on a server in the same site in which the Content Engine server runs. This approach provides a significant savings in time and resources by avoiding multiple round trips across the wide area network (WAN) for content. If your configuration has multiple Content Engine servers, you can create the Image Services cache on any of the Content Engine servers as long as all of the Content Engine servers are in the same physical location. If you have multiple Content Engine cluster member nodes in separate geographic locations, consider creating an Image Services cache in each location. However, you can only create one Image Services cache for each location. *Even if your Content Engine is connected to two separate Image Services domains with CFS-IS, you cannot point a single remote Content Engine to more than one Image Services cache.*

**Content cache:** It is also possible to create a Content Engine content cache at a remote site. For more information, see **Help on FileNet Enterprise Manager Administration tool** → **Content Storage** → **Content Cache Areas**.

### Configuration

To configure remote Image Services caching, we make adjustments to several P8 components.

#### *Application Engine configuration*

The Application Engine installation prompts for the following values:

- RemoteServerUrl
- RemoteServerUploadUrl
- RemoteServerDownloadUrl

You can replace the default values during installation or replace them later in the `WcmAPIConfig.properties` file.

Set the values this way:

- **RemoteServerUrl**: Content Engine server in the primary (home) location
- **RemoteServerUploadUrl**: Content Engine server in the remote location where the Image Services Cache Services Manager cache is located
- **RemoteServerDownloadUrl**: Content Engine server in the remote location where the Image Services Cache Services Manager cache is located

### *Content Engine*

Create an Image Services Cache Services Manager cache in the same site as the Content Engine by following these steps:

1. Install the base Image Services 4.*x* software on the Content Engine server or on another server in the satellite system and follow the instructions in IBM FileNet Image Services Installation and Configuration Procedures, SC19-2680, for the appropriate Content Engine platform to configure a cache-only Image Services Application server. Add the new Image Services Application server to the same IS domain as the remote IS system. If the Content Engine server links to several remote Image Services systems, you need to configure several physical Image Services Application servers, each assigned to the appropriate Image Services domain. Only one instance of Image Services can be configured on a single server. Each Image Services Application server will be able to provide Content Engine with Cache Services Manager services for its own Image Services domain, because all of the Image Services Application servers are on the same local network with the Content Engine server.

   **Restriction**: You cannot install Process Engine software on the same server as the remote Image Services Application server with Cache Services Manager services.

2. Install the appropriate Image Services service packs. Install at least Image Services 4.0 service pack 5 or later.
3. Specify the Cache Services Manager Cache addresses:
   a. Open FileNet Enterprise Manager and view the properties of your Image Services FCD.
   b. Click the **Site Settings** tab.
   c. Select the Site from the drop-down list corresponding to the site where you have installed an Image Services Cache Services Manager cache.
   d. Enter the NCH three-part name for this Cache Services Manager cache. Each part of the name is separated by a colon, and the name takes the form:

      `page_cache<n>:<domain>:<organization>`

There must be no more than one local Cache Services Manager cache name for a given value of *<domain>:<organization>*.

For example, for domain ntvaga and organization FileNet, the names are of the form `page_cache<n>:ntvaga:FileNet`, where *<n>* is an unsigned decimal integer greater than or equal to `2`. The default Cache Services Manager cache on the document server is page cache number 1 (for example, `page_cache1:ntvaga:FileNet`).

This property will be used when the Content Engine server is running within the site identified by this object's Site property.

#### Testing

You can test the configuration by adding a small number of documents to Image Services and, then, retrieving them from the remote location, using either Image Services or P8 tools, or both, depending on your configuration.

### 3.8.3 Image Services capacity planning

CFS-IS can add an additional load on your Image Services system, especially if there will be a bulk import to Content Engine. Be prepared to increase system and Image Services resources to accommodate the additional load. These resources include memory, CPU, database space (IndexDB and MKF Perm DB), and process stubs. Be especially cognizant of DOC, INX, and Cache Services Manager (CSM) stubs.

## 3.9 Best practices

In this section, we describe optional recommendations about how to configure aspects of your CFS-IS system.

### 3.9.1 Schema

Consider these best practices in the areas of document classes, document class consolidation, and database indexes.

#### Document classes

Unlike Image Services, Content Engine allows a hierarchical document class structure, which is especially helpful in federation. A hierarchical document class structure allows you to keep federated Image Services data separate from native P8 data and federated data from other external systems. Hierarchical document class structure also enables the use of property template inheritance.

We recommend creating within the CE object store a single parent document class for all federated Image Services systems. Then, create a subclass for each Image Services system. Then, create a subclass under that subclass for each document class. Each subclass can inherit property templates from the classes above it.

Table 3-7 demonstrates this idea.

*Table 3-7 Property template inheritance*

| Document class | Assigned properties | Inherited properties |
|---|---|---|
| /CFSIS | IS DocID<br>IS Entrydate | N/A |
| /CFSIS/IS1 | N/A | IS DocID<br>IS Entrydate |
| /CFSIS/IS2 | Account number<br>Last name | IS DocID<br>IS Entrydate |
| /CFSIS/IS2/Claims | Claim number<br>Claim date | IS DocID<br>IS Entrydate<br>Account number<br>Last name |
| /CFSIS/IS2/Apps | Application date | IS DocID<br>IS Entrydate<br>Account number<br>Last name |
| /CFSIS/IS2/Apps/Cons | Social security number (SSN) | IS DocID<br>IS Entrydate<br>Account number<br>Last name<br>Application date |
| /CFSIS/IS2/Apps/Comm | Employer Identification Number (EIN) | IS DocID<br>IS Entrydate<br>Account number<br>Last name<br>Application date |

## Document class consolidation

You can map multiple Image Services document classes to a single Content Engine document class. This ability might help you consolidate your existing Image Services document classes and indexes when federating to Content Engine. See Table 3-8 on page 138 for an example.

Archived

*Table 3-8 Many-to-one Image Services class to Content Engine class mapping example*

| IS document class | IS index | Content Engine document class | Content Engine property |
|---|---|---|---|
| Loan | Last_Name | Loans | LastName |
| Loan_Doc | Lname | Loans | LastName |

When mapping multiple Image Services indexes to a single Content Engine property, be sure to define the Content Engine property size to be at least as large as the largest mapped Image Services index.

## Database indexes (retrieval keys)

After you have federated the bulk of your Image Services documents, create indexes on the Content Engine properties that you will use to uniquely search for federated documents. Indexes add overhead to the Content Engine system and can slow the performance of bulk imports of Image Services documents, but they will significantly improve query performance when searching for documents. Follow the same guidelines that you follow when creating Image Services retrieval keys and indexes on other Content Engine properties. Follow these steps to create Content Engine indexes:

1. From IBM FileNet Enterprise Manager, drill down to the document class that you configured with CFS-IS. In our example, we used **Rb_dc_CFSIS**.
2. Right-click the document class, and select **Properties**. Click the **Property Definitions** tab.
3. Click the desired property. In our example, we selected **Policy Number CFSIS**.
4. Click **Edit**.
5. The field that is labeled Indexed indicates whether the property already has an index. If the field value is “not indexed,” create an index by continuing with the rest of the steps. See Figure 3-40 on page 139.

Archived

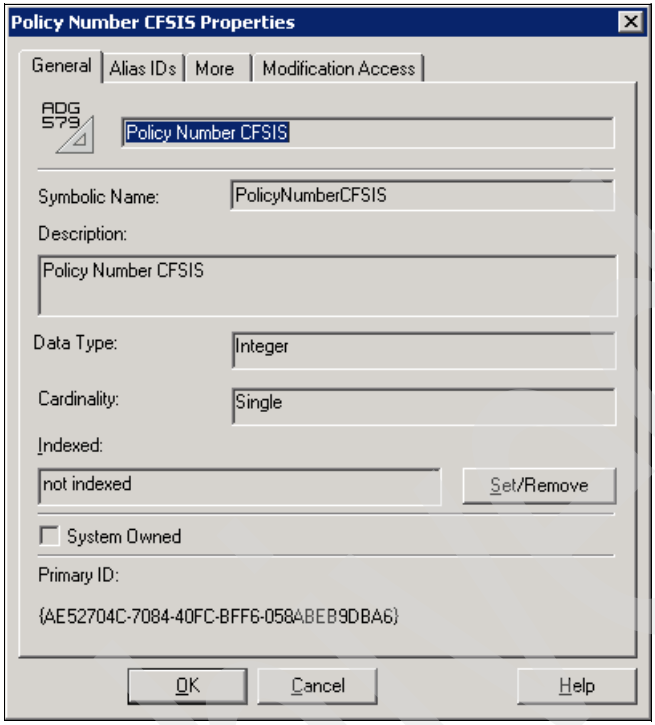



*Figure 3-40 Create property index before it is created*

6. Click **Set/Remove**.
7. Select **Set**.
8. Select **Single Indexed**. See Figure 3-41 on page 140.

Archived

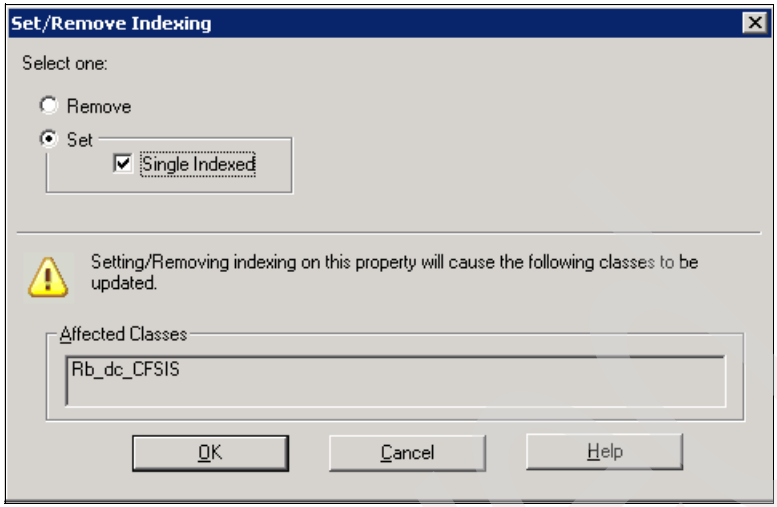



*Figure 3-41 Create property index: Index settings*

9. Click **OK** to exit the Property Definition window.
10. Click **OK** to exit the Document Class window.

## 3.9.2 Marking sets

If you want to retain the Image Services catalog entry for federated documents and continue to modify metadata in Image Services, it is important not to update metadata directly in Content Engine for these documents. Updating the metadata directly will result in out of sync Image Services and Content Engine catalogs, which will prevent the Import Agent from federating future metadata updates for the out of sync documents.

To prevent users from directly updating properties in Content Engine on federated documents, we recommend that you create a temporary marking set for the mapped document classes.

Markings and marking sets are special objects with Content Engine objects:

- *Markings* are objects that combine metadata behavior with access control behavior in a way that allows an object's access control to change by changing a property value.
- *Marking sets* are containers for markings and are associated with a property template that can then be used to add a property to one or more classes.

Markings enable you to control access to objects based on specific property values. When a marking is applied to an object, the resulting access permissions for the object are a combination of the settings of its original access permissions and the settings of the marking's constraint mask for each marking that is applied to it. The result of this combination is the effective security mask.

For CFS-IS, you can define a string property with a specific value and add this property to each Content Engine document class that is mapped to an Image Services document class. This property does not need to be mapped to any Image Services property. This property acts as an anchor for a marking set that will prevent any updates to the Content Engine document class.

After all of the Image Services documents in the class have been federated and the catalog entries have been removed from Image Services, you can remove the marking property from the Content Engine document class. When this property has been removed from all Content Engine document classes, you can delete the marking set.

> **Marking sets**: While this temporary marking set is in place, Enterprise Records cannot declare as records any documents in the affected document class. To declare a document as a record, Enterprise Records needs to update certain ownership and security properties. When the marking set has been removed, Enterprise Records can then successfully declare the documents as records. For more information about markings and marking sets in general, see the FileNet P8 Enterprise-wide Administration help for Security.

In the following steps, we create a marking set, a property template, and assign the property to a document class.

## Create a marking set

Perform the following steps to create a marking set:

1. From IBM FileNet Enterprise Manager, right-click **Marking Sets**.
2. Click **New Marking Set**.
3. Click **Next** on the welcome window.
4. For Marking Set Type, select **List - non-ordered**.
5. For Marking Set Name, assign a name. We chose `CFS_NoUpdate`.
6. Click **New Marking**.
7. For Marking Value, assign a name. We chose `NoUp`.
8. Click the **Constraint Mask** tab.

9. Click **Deselect All**, and then, select only **Modify all properties** and **Create instance**.
10. Click the **Security** tab.
11. Click **Add**, search for and select **#AUTHENTICATED-USERS**, and click **OK**.
12. In the Type field, select **Allow**.
13. In the Rights field, clear the selection from each right. See Figure 3-42.

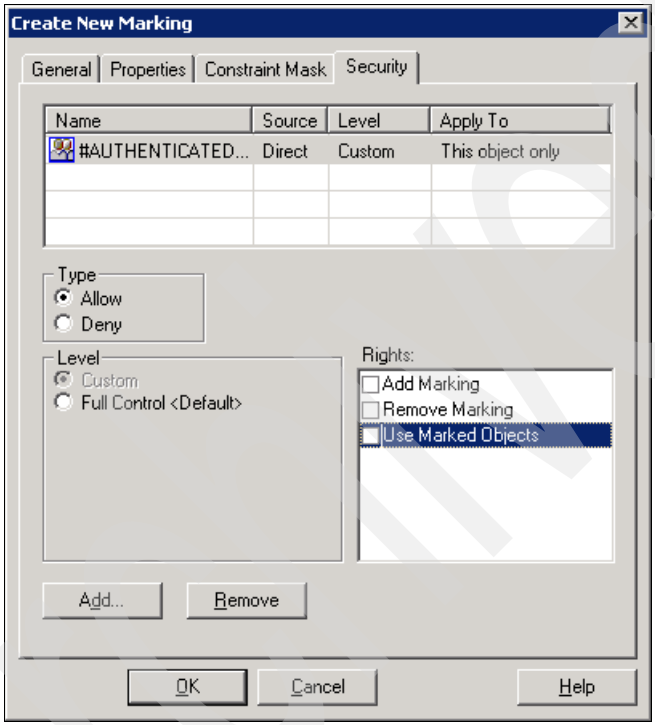



*Figure 3-42 Marking security: #AUTHENTICATED-USERS*

14. Click **Add**, search for and select **Administrator**, and click **OK**.
15. In the Type field, select **Allow**.
16. In the Rights field, clear only **Use Marked Objects**. See Figure 3-43 on page 143.

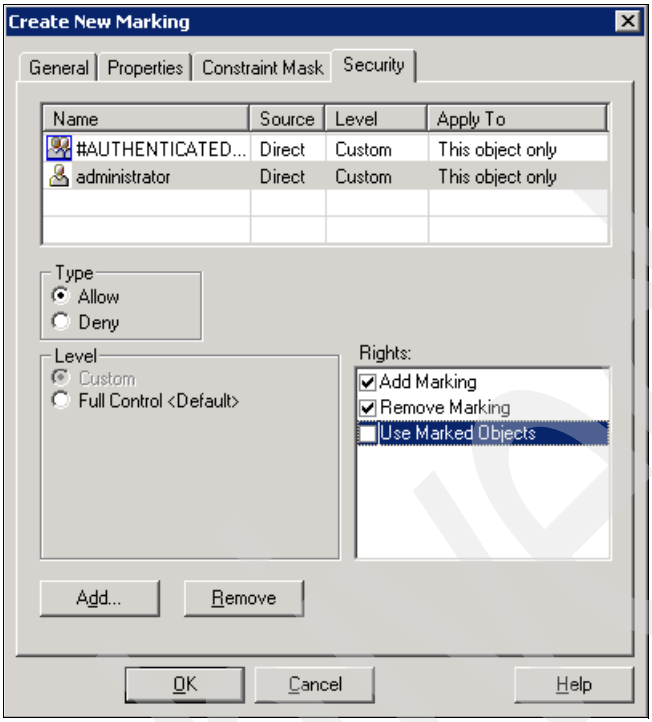



*Figure 3-43 Marking security: Administrator*

17. Click **OK** to finish creating the marking.
18. Click **Next** and click **Finish** to complete the process of creating the marking set. You will see a message that the marking set was created successfully.

## Create a property template

Perform the following steps to create a property template:

1. In IBM FileNet Enterprise Manager, expand the **Object Stores** node.
2. Right-click the desired object store, click **New**, and then, click **Property Template**.
3. Click **Next** on the welcome window.
4. Assign a name to the property template, and click **Next**. In this example, we used the name `CFSNoUpdate`.
5. When it prompts you for a data type, select **String**, and click **Next**. Marking sets can only be linked to string data types.
6. For the choice list and marking set selections, click **Assign marking set**, and select the marking set that we created in the previous step. Click **Next**.

7. When prompted for Single or Multi Value, select **Single**.
8. Click **More.**
9. Select **Value Required**.
10. For Default Value, type the name of the marking that you created in the previous step. In our example, we chose `NoUp`. The value is case sensitive.
11. Set Maximum String Length to the length of your marking. In our example, we typed `4` to match the length of our value of NoUp.
12. Click **OK** to return to the previous window, click **Next,** and then, click **Finish** to complete the process.

### Assign the property to a document class

Perform the following steps to create a marking set and to add it to the mapped document class:

1. From IBM FileNet Enterprise Manager, drill down to and right-click the desired document class. In our example, we used **Rb_dc_CFSIS**.
2. Select **Add Properties to Class**.
3. Click **Next** on the welcome window.
4. From the list of available properties, select the property that you created in the last step, and click **Add**. In our example, we used **CFSNoUpdate**. Then, click **Next**.
5. When prompted to set property attributes, click **Next**.
6. Click **Finish**.

Now that the marking set is in place, Content Engine will prevent metadata updates to documents in the affected document classes, for non-administrator users. Administrators can still make updates.

> **Note:** Now that marking sets are in place, documents in the affected class cannot be declared as records by Enterprise Records, except by a system administrator.

Archived

4

# Content Federation Services for Content Manager OnDemand

IBM FileNet P8 Content Federation Services (CFS) for IBM Content Manager OnDemand (CFS-CMOD) provides the ability to federate documents from IBM Content Manager OnDemand and view the federated documents in IBM FileNet Content Manager. This chapter discusses how to implement CFS-CMOD effectively in your Enterprise Content Management (ECM) environment.

We cover the following topics in this chapter:

- Architecture
- Federation process
- Planning
- Installation and configuration
- Federate CMOD documents to FileNet P8
- View federated CMOD documents in WorkplaceXT
- Federated records management with CFS-CMOD
- Troubleshooting

© Copyright IBM Corp. 2010. All rights reserved.

# 4.1 Architecture

The CFS-CMOD architecture that is described in this section expands on the CFS conceptual model that is discussed in Chapter 2, “Content Federation Services architecture” on page 33. We discuss the specific manner in which the CFS conceptual model is implemented by the CFS-CMOD offering.

## 4.1.1 Mapping of CMOD data model to P8

Before discussing the architectural components of the CFS-CMOD implementation, we briefly describe how the CMOD data model maps to the P8 data model.

CMOD inserts documents into its repository in large batches. This large batch of documents is called a *load*. All the documents in the load are added to a CMOD *application group*. A CMOD repository can have multiple application groups. The metadata properties of CMOD documents are called *fields*.

Thus, the CMOD data model maps to the P8 data model in the following manner:

- A CMOD application group corresponds to a P8 document class.
- A CMOD document corresponds to a P8 document.
- The fields of a CMOD document correspond to the properties of a P8 document.
- A CMOD document always maps to a single content element in a P8 document.
- A P8 object store can contain multiple document classes. Hence, it can contain mappings to multiple CMOD application groups for that object store.
- A CMOD application group cannot be mapped to more than one P8 object store.

## 4.1.2 Components

Figure 4-1 on page 147 shows the CFS-CMOD components that make up the CFS conceptual model.

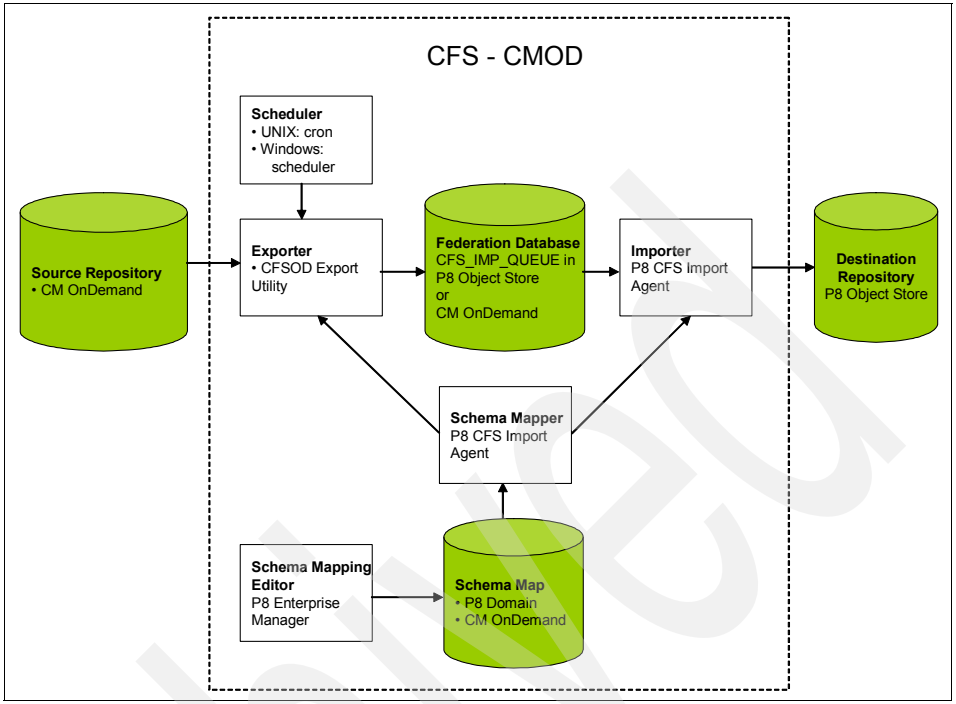



*Figure 4-1 CFS-CMOD components that make up the CFS conceptual model*

The following components make up CFS-CMOD:

- Schema mapping editor
- Schema map
- Scheduler
- Source repository
- Exporter
- Federation database
- Importer
- Schema mapper

## Schema mapping editor

To prepare CMOD documents for federation, you create a schema map to map CMOD application groups and fields to Content Engine document classes and properties. This mapping occurs in Content Engine using the IBM FileNet Enterprise Manager.

## Schema map

The results of the schema mapping process are persisted in both the P8 domain and the CFS_REPOSITORY_MAP table of the OnDemand database.

### Scheduler

The CFSOD exporter utility is scheduled to run using the operating system scheduling utility. For UNIX systems, cron is used. For Windows systems, the Windows Scheduler is used. A typical schedule runs the utility once a day, when the expected system utilization is low.

### Source repository

The source repository for CFS- CMOD is the OnDemand database.

### Exporter

The exporter for CFS-CMOD is the CFSOD exporter utility, which is a stand-alone application that connects to the OnDemand database and the CFS federation database. The CFSOD exporter utility performs queries against the OnDemand database for documents to federate and creates the federation requests in the federation database that are processed by the CFS Import Agent.

### Federation database

The federation database is the CFS_IMP_QUEUE table that is created in either the OnDemand database or in the Content Engine object store database. The CMOD exporter creates federation requests and inserts them into this table. The CFS importer then processes these requests to federate these CMOD documents and removes them from this table.

### Importer

The importer for CFS-CMOD is the CFS Import Agent that runs as part of the Content Engine.

### Schema mapper

The schema mapper for CFS-CMOD is also the CFS Import Agent that runs as part of the Content Engine.

## 4.2 Federation process

The CFS-CMOD federation process involves two distinct phases that are performed on the CMOD server and the Content Engine server: exporting and importing.

Archived

### Exporting

The exporting phase occurs wherever the CFSOD exporter utility is run. We recommend that you run this utility on the same server that hosts this database, because the exporter interacts directly with the federation database.

When you configure the CFS option for a CMOD application group, you can cause all existing documents and new documents to be federated automatically. Or, you can choose to manually specify which documents to federate through an OnDemand client. Both cases cause the OnDemand server to add these selected documents to the ARSCFSODWORK table in the OnDemand database.

When the CFSOD exporter utility is run, it processes the entries in the ARSCFSODWORK table, formats them into CFS federation request records, and adds these records to the federation database.

### Importing

The importing phase of CFS-CMOD occurs within Content Engine. The CFS Import Agent, if enabled, runs continuously in the Content Engine server. It periodically queries the federation database for requests that are inserted by the CFSOD exporter utility. The CFS Import Agent processes document create, update, and delete requests and performs the corresponding action on the associated Content Engine document. If the processing completes successfully, the requests are removed from the federation database.

## 4.2.1 Federated document operations

You can perform several operations to manage federated CMOD documents, including create, retrieve, lockdown, and delete.

### Create

The create operation is performed by the CFS Import Agent. The following list describes the details of this operation:

- CMOD documents federated to Content Engine cause a corresponding Content Engine document to be created with the mapped properties of the CMOD document.
- The Content Engine document contains a reference to the content of the CMOD document that is located on the CMOD server.

Archived

## Retrieve

You can retrieve the native bytes of CMOD documents using any application that is written to the P8 API. However, due to the nature of the document types that are stored in the CMOD database (such as Advanced Function Presentation (AFP), Mixed Object Document Content Architecture (MO:DCA), and line data), you can only view these documents using WorkplaceXT.

## Update

CMOD metadata does not change, so there are no metadata updates with CFS-CMOD.

## Lockdown

To prevent updates by CMOD native applications to documents that have been federated to Content Engine, P8 applications, such as IBM Enterprise Records (formerly known as IBM FileNet Records Manager), can initiate a lockdown of the document on the CMOD server. The following list describes the details of this operation:

- CMOD can be configured to automatically lock down documents for an application group when they are federated. This capability gives CFS control of these documents so that they cannot be deleted unless a delete request is received from Content Engine using an application, such as FileNet Enterprise Manager or WorkplaceXT.
- If this option is not enabled, individual documents in CMOD can still be locked down when declared as records by Enterprise Records.
- Lockdown causes a retention hold to be placed on the CMOD documents, which prevents them from being deleted by native OnDemand applications, such as the OnDemand auto-delete sweep process.

**Performance consideration:** Always consider the extra load that is added to the existing system when working with records. For instance, a lockdown operation, when performed on a large number of federated documents, can consume significant system resources and needs to be considered. Records Administrators frequently searching for records can also be costly. The Enterprise Records sweep processes (disposition and hold) can also greatly increase the system load. Plan ahead to insure that you have sufficient hardware to handle your operations and that you have sufficient time to complete these operations.

Archived

### Delete

The delete operation of CFS-CMOD is bidirectional. The following list describes the details of this operation:

- Deletes of CMOD documents by CMOD native applications are propagated to the Content Engine where the corresponding Content Engine document is deleted.
- Deletes from P8 applications of federated CMOD documents are propagated to CMOD where the corresponding CMOD document is deleted.

## 4.3 Planning

There are many tips and best practices for designing a successful OnDemand solution. Similarly, when planning for Content Federation Services with IBM FileNet Content Manager and IBM Content Manager OnDemand, several tips and best practices exist for designing a successful CFS-CMOD solution, as well. In this section, we outline several design best practices and tips to help you learn how to design an efficient CFS-CMOD solution.

### 4.3.1 When to use Content Federation Services for Content Manager OnDemand

When considering when to use CFS-CMOD, you need to ask yourself what kind of business problem you are trying to solve. It is important to answer this question honestly and correctly, which helps you to determine when to use CFS-CMOD.

#### Compliance

Probably the strongest reason for using CFS-CMOD is to comply with a regulation. For example, an insurance organization might need to retain each customer's insurance policy for eight years and, then at the end of the retention period, insure that the information is destroyed. The retention management available within CMOD can handle this simple use case. What happens if the required retention period changes and must be applied retroactively to loads that have already been processed? What if a specific document within a load needs to be held longer than the original retention period because of a lawsuit?

CFS-CMOD provides the ability to lock down CMOD documents to prevent their deletion even after the retention date for the associated CMOD load has been reached. After CMOD documents are federated to an object store, they can be declared as records. The record declaration causes the documents to be locked down until the end of the appropriate records disposition cycle.

Archived

### Single user interface

Another reason for using CFS-CMOD is to unify the access across P8, CMOD, and Image Services. For example, a company stores all of their electronic documents in Image Services. However, the company is planning to phase out the current IS solution to move to OnDemand, because the ingestion volume is due to increase significantly in the next 12 to 18 months. During the transition period, new documents are stored in OnDemand and must be retrieved in P8. The existing documents in Image Services still need to be accessible in P8, as well.

CFS-CMOD is a seamless and easy way to extend the existing enterprise catalog. Having the ability to maintain access to IS and OnDemand documents by way of a single interface helps you to develop or implement new processes while not disrupting the current processes.

## 4.3.2 Best practices

The following best practices are suggestions, but they are not comprehensive. These best practices are the most useful practices for designing a CFS-CMOD solution:

- Mapping OnDemand fields to the Content Engine properties
- Running the CFSOD exporter utility on the Content Engine server
- Deciding when to run the exporter
- Increasing the CFS import performance

### Mapping OnDemand fields to the Content Engine properties

The federated CMOD documents are filed in the Unfiled Documents folder in the Content Engine after federation. Therefore, you cannot use a predefined application, such as WorkplaceXT, to browse to the federated documents. Instead, you must query for the documents using the WorkplaceXT search function. So, ask how the users want to find the federated documents.

Significant analysis is needed to learn how the federated documents are accessed in the Content Engine server. Understanding how the federated documents are accessed in the Content Engine server helps you determine exactly which fields need to be mapped.

In general, the fields that exist in an OnDemand application group need to be mapped to the properties in the Content Engine server to support the same OnDemand queries in P8. For example, if the OnDemand users use the account number field to find the documents in the OnDemand server, map that account number field to an account number custom property in the Content Engine server so that you can perform the same search in P8.

### Running the CFSOD exporter utility on the Content Engine server

When federating CMOD documents, the CFSOD exporter utility creates more database transactions against the CFS importer database tables than against the OnDemand database tables. Keeping these transactions local to the CFS importer database tables improves the federation performance. Therefore, the CFS importer database needs to be created on the same database server that hosts the Global Configuration Data (GCD) and object store databases. Also, when creating the Java Database Connectivity (JDBC) data sources for the CMOD fixed content device, the data sources need to point to this local CFS importer database. See “Configure JDBC data sources for the fixed content device” on page 165 for details about creating the JDBC data sources.

The CFSOD exporter utility ships with the OnDemand Web Enablement Kit. Because installing the OnDemand Web Enablement Kit is a requirement for configuring CFS-CMOD on the Content Engine server, it makes sense to run the CFSOD exporter on the Content Engine server. See 4.5.4, “Configure and run the CFSOD exporter utility” on page 173 for more details about configuring the exporter.

### Deciding when to run the exporter

You must decide when and how often to run the exporter. Determine the number of documents that your organization will federate. Also, analyze the exact size of each load for your organization.

In general, you must run the exporter frequently enough so that the rows in the ARSCFSODWORK table on the OnDemand server do not occupy too much space.

**Guideline:** Each row in the ARFCFSODWORK table can have up to 10 MB of compressed binary large objects (BLOB).

However, we recommend that you do *not* run the exporter immediately after a load completes. For example, when you load one million documents overnight and then realize the next day that something was wrong with the load, normally, you must unload this bad load from the OnDemand server. At this point, if you have already run the exporter and federated this bad load, it is a problem to clean up, because it takes a long time to delete one million documents from the Content Engine. This situation is even worse if these federated documents were declared as records automatically in P8, because after the source CMOD documents are locked down, you can only delete both federated and source OnDemand documents using Enterprise Records. Therefore, you must delete

the records, the federated documents, and finally, the source CMOD documents, which takes even longer to clean up everything.

You can use the time offset option (the -D option) in the CFSOD exporter utility to make sure that the load that you federate is a good load. The time offset option allows you to federate only the rows in the ARSCFSODWORK table that have an earlier time stamp. For example, if you load one million documents every day, you can set the time offset to federate Monday's load on Friday. That way, if any of the loads are bad during the week, you can simply use the **arsadmin unload** command to remove the bad loads before they are actually federated.

We also recommend that you run the CFSOD exporter utility when the system is not busy. Do not run the exporter while you load millions of documents or delete documents using the **arsmaint** command. The exporter, arsload, and arsmaint all access the OnDemand database tables; therefore, you want to avoid these programs competing for database locks. You need to allot enough time to complete the federation before loading a new load or before arsmaint is scheduled to run.

### Increasing the CFS import performance

When federating CMOD documents in extremely high volumes, you can achieve better federation throughput by dropping the I_EXTERNALIDENTI76 index on the EXTERNALIDENTITY table in the Content Engine server.

The EXTERNALIDENTITY table maps the Content Engine object ID of the federated CMOD document to the document ID of the source CMOD document. WorkplaceXT uses this table to retrieve and view the content from the OnDemand server by getting the CMOD source document ID by querying for the Content Engine object ID in this table.

The I_EXTERNALIDENTI76 index indexes the CMOD source document ID on the table; however, it is not used by WorkplaceXT or the Content Engine. As a result, you can safely remove this index to increase the insert operation on the EXTERNALIDENTITY table. Also, removing this index does not affect the Content Engine performance.

## 4.4 Installation and configuration

In this section, we describe the following installation and configuration tasks for CFS-CMOD:

1. Install IBM Content Manager OnDemand.
2. Install IBM FileNet Content Manager.
3. Configure IBM Content Manager OnDemand for CFS-CMOD.

4. Configure IBM FileNet Content Manager for CFS-CMOD.
5. Federate CMOD documents to FileNet P8.

### 4.4.1 Install IBM Content Manager OnDemand

Installing IBM Content Manager OnDemand is out of the scope for this book. Obtain the complete installation procedure at this Web site:

http://publib.boulder.ibm.com/infocenter/cmod/v8r4m1//index.jsp

### 4.4.2 Install IBM FileNet Content Manager

Installing IBM FileNet Content Manager is out of the scope for this book. Obtain the complete installation procedure at this Web site:

http://www.ibm.com/support/docview.wss?rs=3273&uid=swg27010422

### 4.4.3 Configure IBM Content Manager OnDemand for CFS-CMOD

After installing and configuring the IBM Content Manager OnDemand server, you configure IBM Content Manager OnDemand for CFS in the following manner:

1. Enable CFS on IBM Content Manager OnDemand.
2. Enable CFS in a CMOD application group.
3. Enable the folder to display the locked down documents.

#### Enable CFS on IBM Content Manager OnDemand

In order to use the CFS feature in IBM Content Manager OnDemand, you must activate the CFS-CMOD feature. On Windows, you activate this feature by selecting **Enable CFS-OD** in the Server (Advanced Options) window. See Figure 4-2 on page 156.

Archived

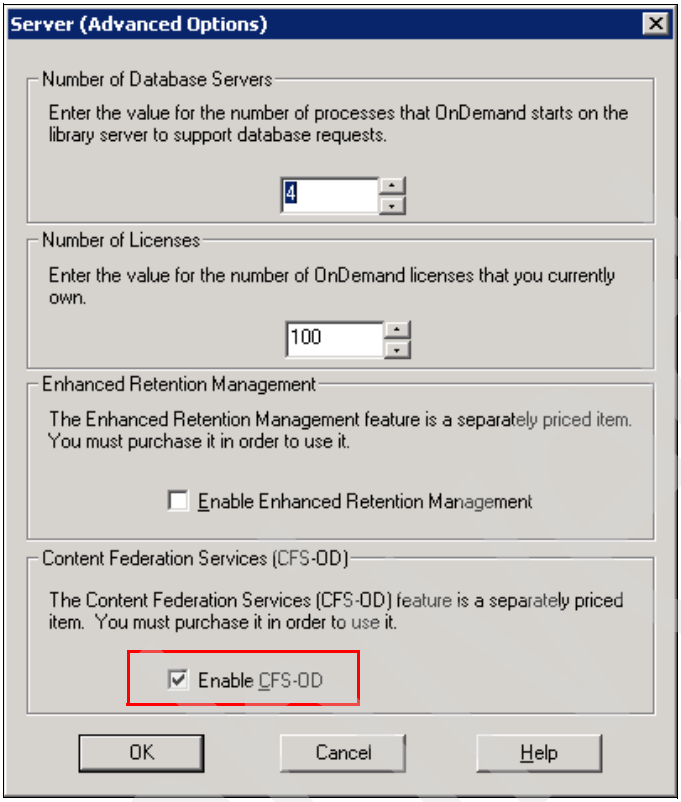



*Figure 4-2 Server (Advanced Options) window to enable CFS-OD*

In other platforms, you activate this feature by adding the following line in the `ars.cfg` file.

```
ARS_SUPPORT_CFSOD=1
```

## Enable CFS in a CMOD application group

After enabling the OnDemand server for CFS, you need to enable an application group for CFS. Follow these steps to enable an application group for CFS:

1. Launch the OnDemand Administration Tool and log in to the OnDemand repository.
2. Expand **Application Groups**.
3. Select an application group that is used for federation from the right side panel.
4. Right-click the application group, and select **Update**.
5. Select the **Field Definition** tab**.**

6. Enter `cfsod` in the Database Field Name, and click **Add**. See Figure 4-3.

> **Important:** The database field name must be *cfsod*. Federation does not work properly if another name is used.

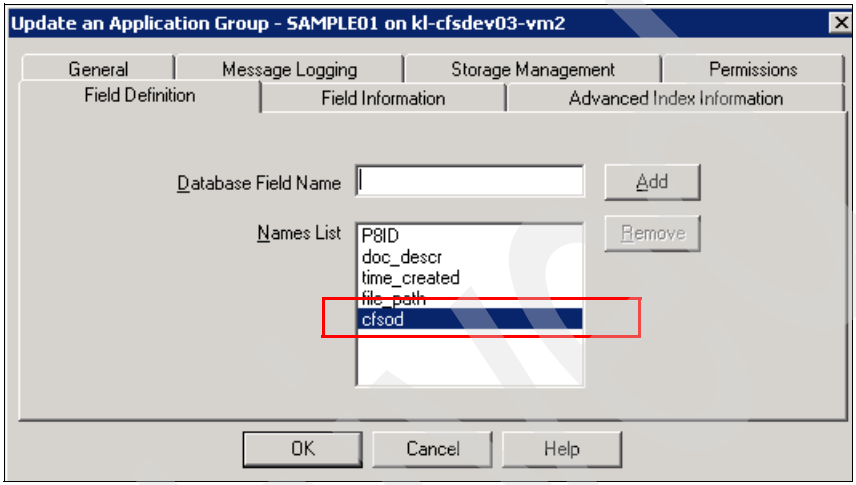



*Figure 4-3 Adding the cfsod database field*

> **Examples:** We use the SAMPLE01 application group throughout this chapter for various examples.

7. Select the **Field Information** tab.
8. Select the **cfsod** field from the Name drop-down menu.
9. Select **Small Int (2)** from the Data Type drop-down menu.
10. Select **CFS-OD**. See Figure 4-4 on page 158.

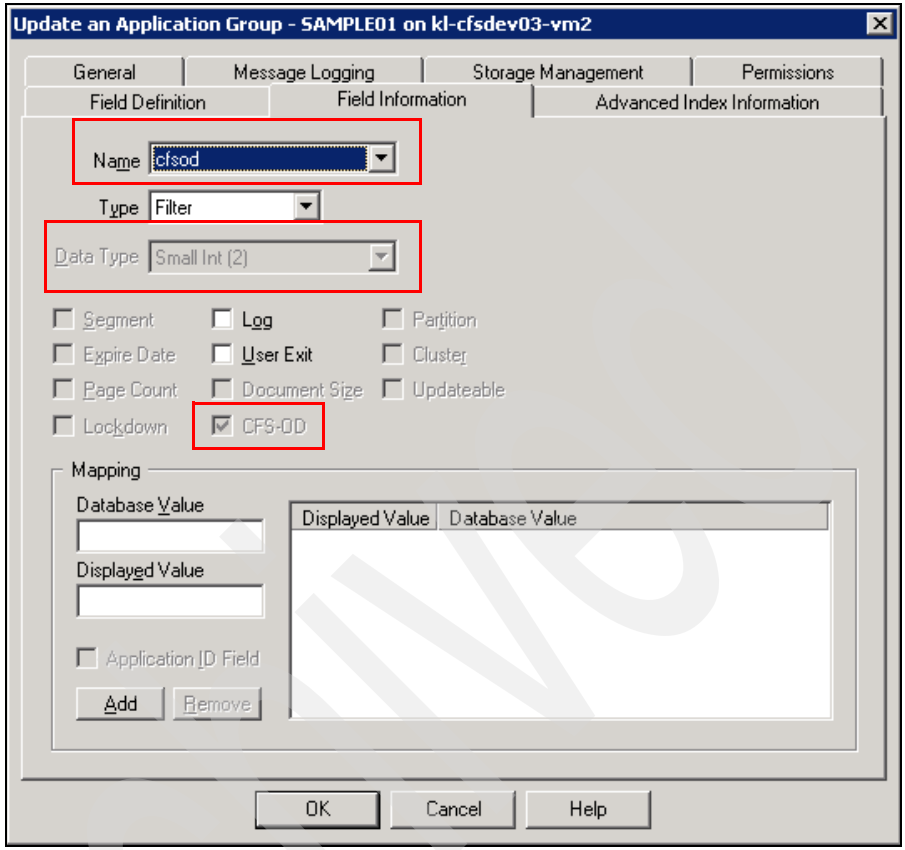



*Figure 4-4 Update an Application Group: Field Information*

11. Select the **General** tab, and click **Advanced**.
12. In the Interoperate with FileNet P8 Platform panel, select **Yes** for “Use Content Federation Services (CFS-OD)?” and select the appropriate check boxes for your needs. See Figure 4-5 on page 159.

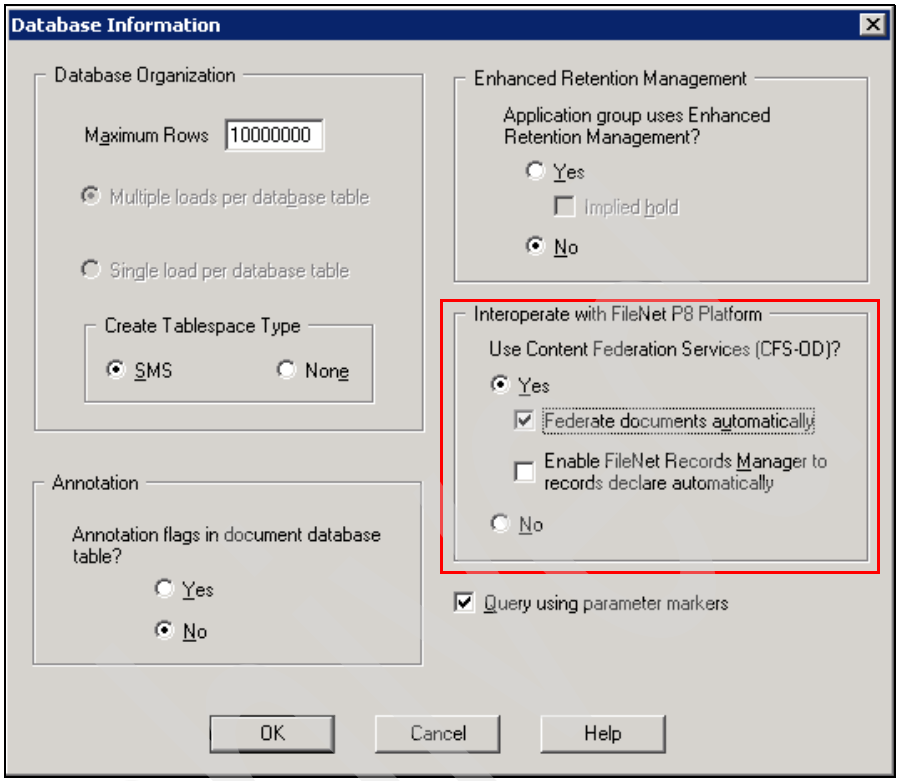



*Figure 4-5 Database Information window*

If you select the “Federate documents automatically” option only, the source CMOD documents are federated during the load, but they are not locked down using CMOD retention holds. You must use Enterprise Records to declare the federated CMOD documents as records to lock down the source CMOD documents.

If you select both the “Federate documents automatically” check box and the “Enable FileNet Records Manager to records declare automatically” check box, the documents are federated and locked down simultaneously using a CMOD retention hold. You still need to use Enterprise Records to declare federated CMOD documents as records, which puts the source CMOD documents under the control of Enterprise Records.

If you do not select any check boxes, the source CMOD documents are not federated automatically, and CMOD retention holds are not placed on the source documents. You can select and manually federate the source CMOD documents individually by using the OnDemand client application or by using the OnDemand **`arsdoc`** command.

13. Select the **Storage Management** tab and make sure that the Expiration Type is set to **Load** in the Life of Data and Indexes panel. See Figure 4-6.

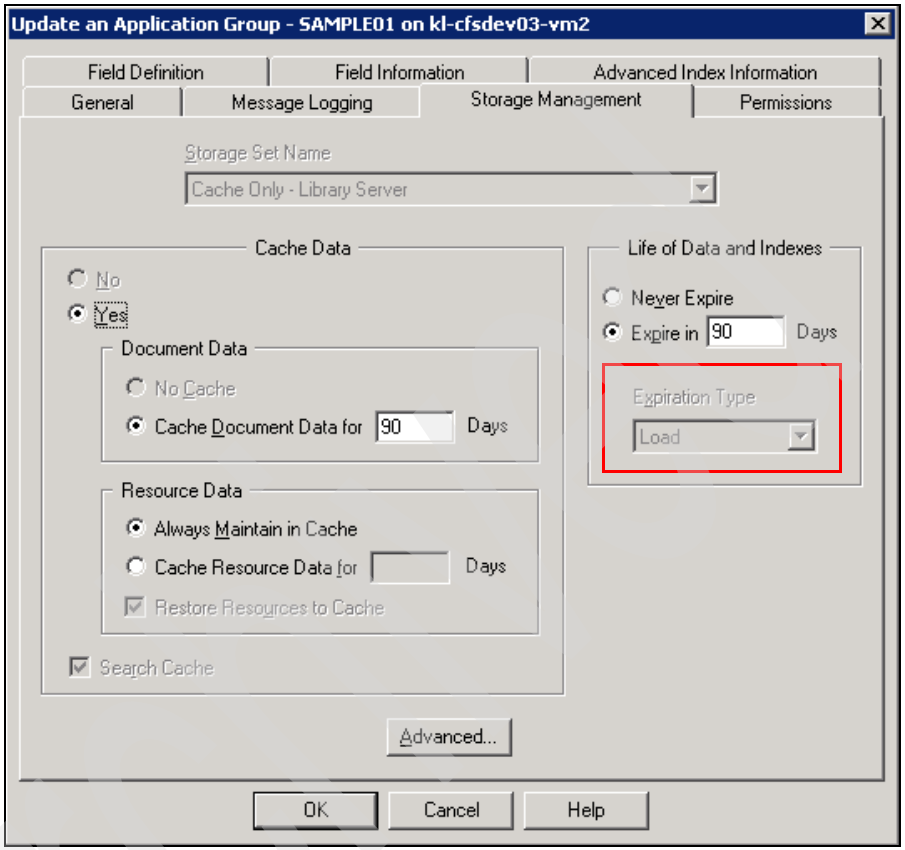



*Figure 4-6 Update an Application Group: Storage Management*

14. Select the **Permissions** tab. Select a user from the Users/Groups list, and click **Add.** Make sure that **CFS-OD** is selected for the selected user. See Figure 4-7 on page 161.

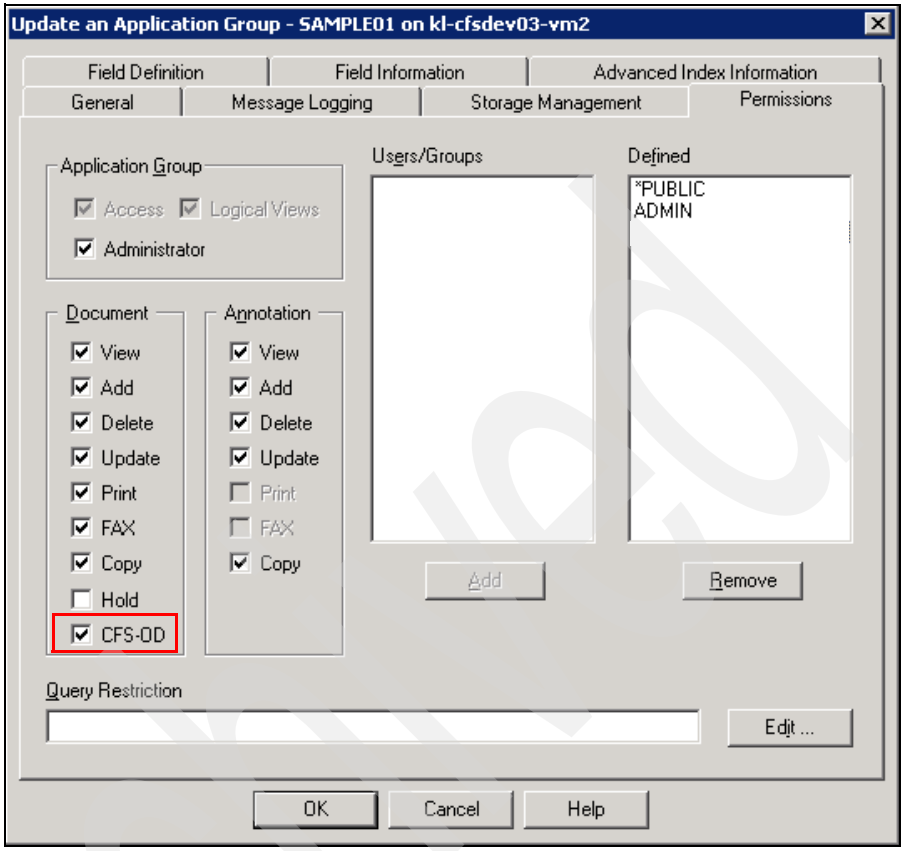



*Figure 4-7 Update an Application Group: Permissions*

## Enable the folder to display the locked down documents

You can enable an OnDemand folder to indicate whether the source CMOD documents have been locked down. Follow these steps to configure the folder:

1. Launch the OnDemand Administration tool, and log in to the OnDemand repository using the admin credentials.
2. Expand **Folders**.
3. Right-click the folder that is associated with the application group that is enabled for CFS-CMOD, and select **Update**.
4. Select **Display Document Hold**. See Figure 4-8 on page 162.

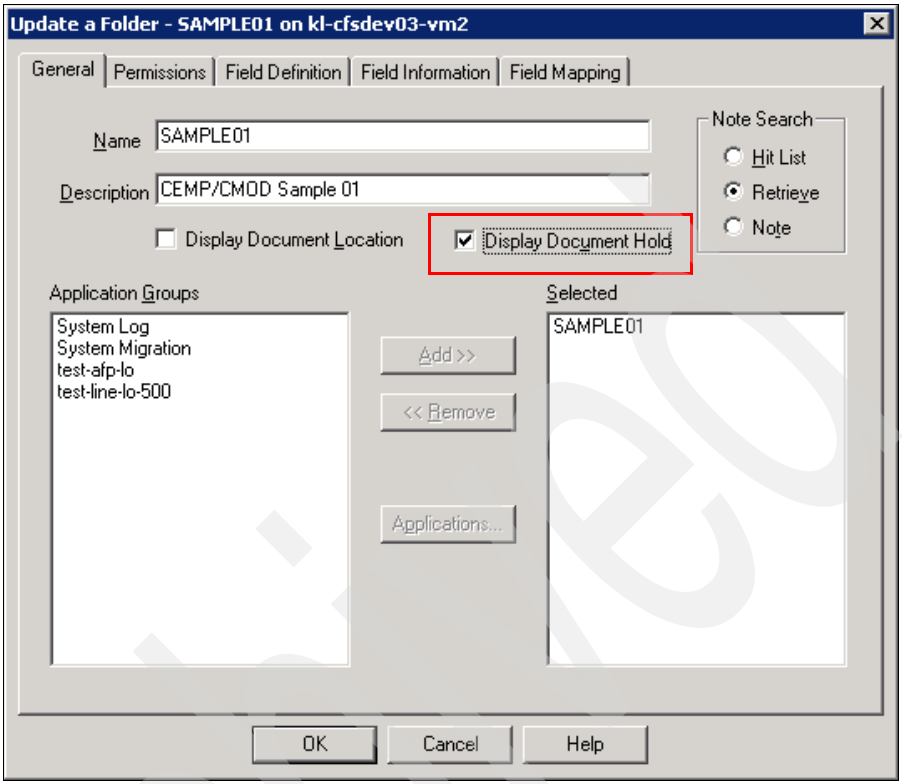



*Figure 4-8 Update a Folder*

## 4.4.4 Configure IBM FileNet Content Manager for CFS-CMOD

Follow these steps to configure IBM FileNet Content Manager for CFS-CMOD:

1. Install the OnDemand Web Enablement Kit.
2. Configure JDBC data sources for the fixed content device.
3. Create the CMOD fixed content device.
4. Create the CMOD fixed storage area.

### Install the OnDemand Web Enablement Kit

The Content Engine uses the OnDemand Web Enablement Kit to access the CMOD repositories. You must install the OnDemand Web Enablement Kit on each Content Engine that is used for CFS-CMOD. The minimum OnDemand Web Enablement Kit version that is required for CFS-CMOD is 8.4.1.0.

**z/OS®:** If your OnDemand server runs on z/OS, the minimum OnDemand Web Enablement Kit version that is required for CFS-CMOD is 8.4.1.1.

Refer to the IBM Content Manager OnDemand Version 8.4.1 Information Center for more information about installing the OnDemand Web Enablement Kit:

http://publib.boulder.ibm.com/infocenter/cmod/v8r4m1/index.jsp

After the OnDemand Web Enablement Kit is installed, you must configure the application server that runs the Content Engine to support the OnDemand Web Enablement Kit.

**Requirements:** In this example, we show you how to configure the OnDemand Web Enablement Kit using WebSphere® on Windows. However, both the OnDemand Web Enablement Kit and IBM FileNet Content Manager support various operating systems and application servers.

For the OnDemand Web Enablement Kit hardware and software requirements, see this Web site:

http://www.ibm.com/support/docview.wss?rs=129&uid=swg27012104

For IBM FileNet Content Manager hardware and software requirements, see this Web site:

http://www.ibm.com/support/docview.wss?rs=3273&uid=swg27010422

If you plan to or already use another application server and operating system, you must perform these tasks:

- Modify the appropriate application server startup script file to include the full path of the OnDemand Web Enablement Kit installation folder to the operating system PATH and LIBPATH (or your operating system's equivalent) environment variables.
- Enter the full path of the `ODApi.jar` file in the Java classpath.

Follow these steps for configuring the OnDemand Web Enablement Kit in the Content Engine application server:

1. Log in to the WebSphere Administrator Console.
2. Navigate to **Application Server** → ***server1*** → **Process Definition** → **Java Virtual Machine**.
3. Enter the full path of the `ODApi.jar` file in the Java classpath. See Figure 4-9 on page 164.

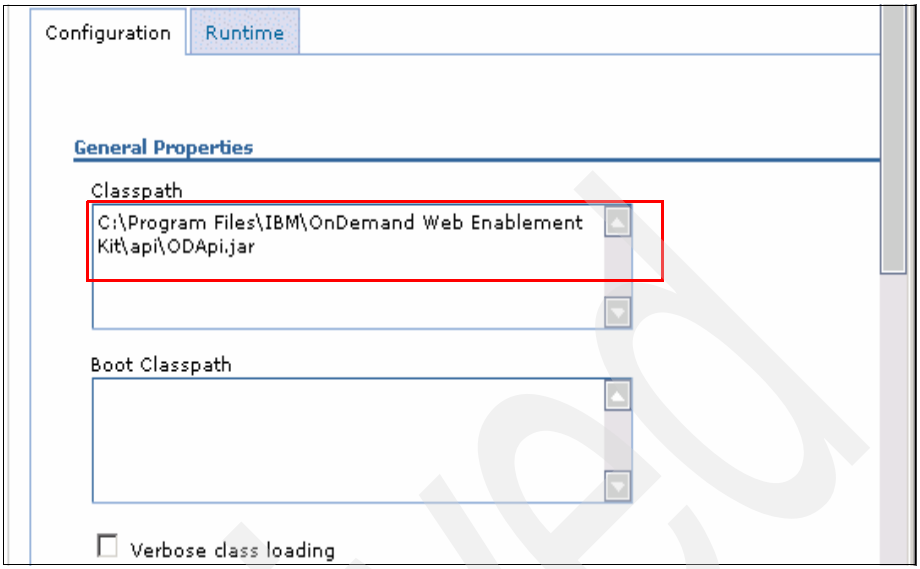



*Figure 4-9 WebSphere Application Server Administrator Console*

Typically, the `ODApi.jar` file is located at the following location:

– In Windows:

  `C:\Program Files\IBM\OnDemand Web Enablement Kit\api\ODApi.jar`

– In UNIX:

  `/usr/lpp/ars/www/api/ODApi.jar`

4. Save your changes to the master configuration.
5. Restart WebSphere.
6. Add the full path of the OnDemand Web Enablement Kit installation directory to the operating system PATH environment variables. See Figure 4-10.

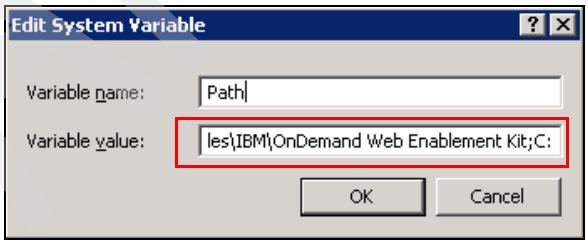



*Figure 4-10 Adding the OnDemand Web Enablement Kit installation directory to the PATH environment variable*

## Configure JDBC data sources for the fixed content device

Before you can create the CMOD fixed content device, you must create two Java Database Connectivity (JDBC) data sources in the Content Engine application server (see Figure 4-11 for our example). These two data sources must point to the database where the CFS importer tables are created. The actual importer tables are created when a CMOD fixed content device is created. Refer to “Create the CMOD fixed content device” on page 165 for more information about how to create a CMOD fixed content device.

> **CFS importer database:** You can create the CFS importer database in either the Content Engine server or the OnDemand server. However, we recommend that you create the CFS importer database in the same database server that hosts the GCD and object store databases.

| Select | Name | JNDI name | Scope | Provider | Descrip |
|---|---|---|---|---|---|
| You can administer the following resources: | | | | | |
| ☐ | CMODDS | CMODDS | Cell=cm-cfsrbNode01Cell | JDBC provider for DB2 | CEMP DataSo |
| ☐ | CMODDSXA | CMODDSXA | Cell=cm-cfsrbNode01Cell | JDBC provider for DB2 (XA) | CEMP DataSo (XA) |

*Figure 4-11 JDBC data sources for the fixed content device*

Refer to the IBM FileNet Content Manager product documentation for more information about how to create JDBC data sources:

http://www.ibm.com/support/docview.wss?rs=3278&uid=swg27010422

## Create the CMOD fixed content device

The CMOD fixed content device is a Content Engine object that points to a CMOD repository. Follow these steps to create the CMOD fixed content device:

1. Launch Enterprise Manager and log in as the administrator user.
2. Right-click **Fixed Content Device**, and select **New Fixed Content Device**.
3. Click **Next** on the welcome window.
4. Enter a name for the fixed content device, and click **Next**.

Archived

5. Enter the following settings for your fixed content device (see Figure 4-12 for our example):
   - **Type**: IBM Content Manager OnDemand
   - **Federation JNDI Data**: The data source name that was created in “Configure JDBC data sources for the fixed content device” on page 165
   - **Federation JNDI XA Data**: The XA data source name that was created in “Configure JDBC data sources for the fixed content device” on page 165
   - **CMOD Server Name**: The OnDemand server name that is shown in the OnDemand Configurator
   - **CMOD User Name**: The CMOD administrator user
   - **CMOD Password**: The CMOD administrator user’s password

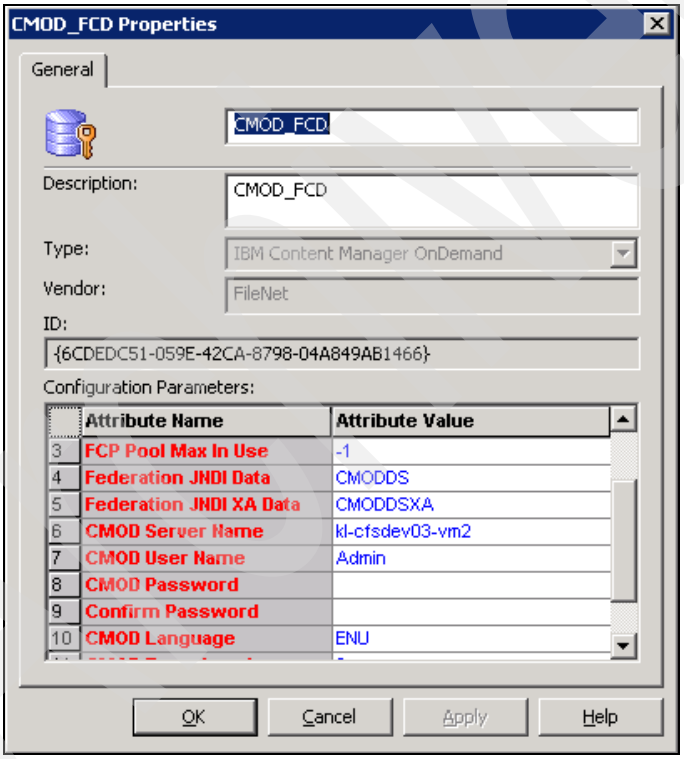



*Figure 4-12 Fixed content device properties*

6. Click **Test Connection** to verify that the fixed content device can connect to the OnDemand server successfully.

7. Click **Next**.
8. Click **Finish**.

## Create the CMOD fixed storage area

A fixed storage area is required for importing federated CMOD documents to the Content Engine. Follow these steps to create a fixed storage area:

1. Create a staging area directory for the fixed storage area (see Figure 4-13). This directory can be either local or remote to the Content Engine.

> **Staging area:** In this example, the staging area is created on a local drive for demonstration purposes. However, if the Content Engine server is running in a clustered environment, the staging area needs to be a shared directory on a local or remote server, which must be accessible by the Content Engine servers.

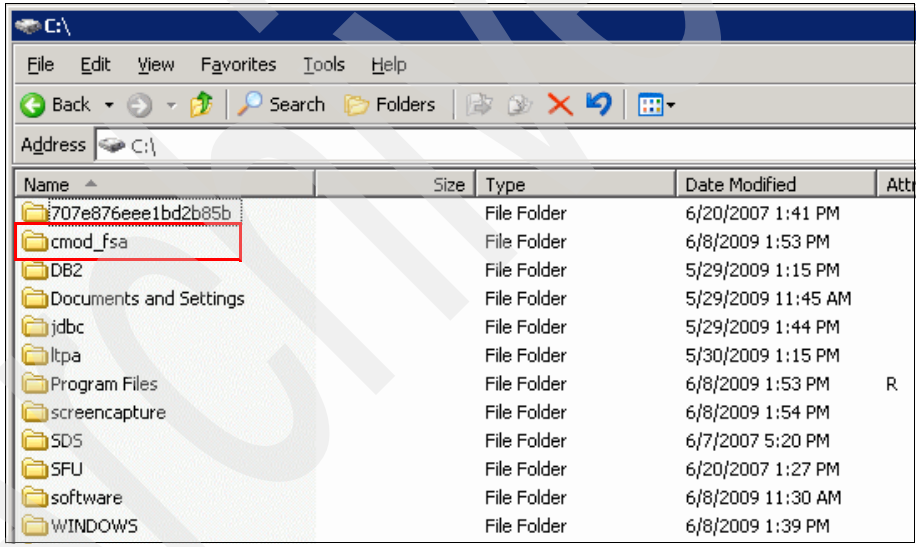



*Figure 4-13 Create a staging folder for the CMOD fixed storage area*

2. Launch Enterprise Manager, and expand **Object Store** → ***<object store name>*** → **Storage Areas**.
3. Right-click **Storage Areas**, and select **New Storage Area**.
4. Click **Next** on the welcome window.
5. Select a site from the list of available sites, and click **Next**.
6. Enter a name for the new storage area, and click **Next**.

7. Select **Fixed Storage Area** for the storage area type, and click **Next**.
8. Select the CMOD fixed content device that was created in “Create the CMOD fixed content device” on page 165, and enter the full path of the staging area that was created in step 1. See Figure 4-14 for our example.

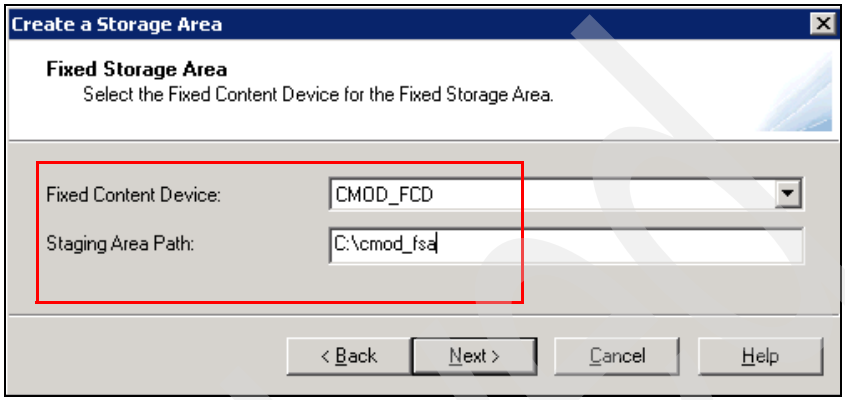



*Figure 4-14 Create a fixed storage area*

9. Click **Next** on the Select the Size Parameters of the Storage Area window.
10. Click **Next** on the Specify Storage Policy window.
11. Click **Finish** to complete the Create a Storage Area wizard.

## 4.5 Federate CMOD documents to FileNet P8

To federate CMOD documents to IBM FileNet Content Manager, perform these steps:

1. Create a custom document class in the Content Engine.
2. Add properties to the new custom document class.
3. Map Content Engine properties to CMOD application group fields.
4. Configure and run the CFSOD exporter utility.

### 4.5.1 Create a custom document class in the Content Engine

Although it is not absolutely necessary, we recommend creating a custom document class that corresponds to a CFS-enabled application group. This design is a good practice, because we can only query for the federated CMOD documents in WorkplaceXT. All of the federated CMOD documents are filed to the Unfiled Documents folder in the Content Engine. WorkplaceXT cannot navigate to the documents filed in the Unfiled Documents folder. By creating a

custom document class and mapping the CMOD fields to the Content Engine properties, it makes it easier to query for the federated CMOD documents in WorkplaceXT.

Assume that the SAMPLE01 application group, as shown in Figure 4-15, is a CFS-enabled application group and that we want to create a custom document class that corresponds to this application group.

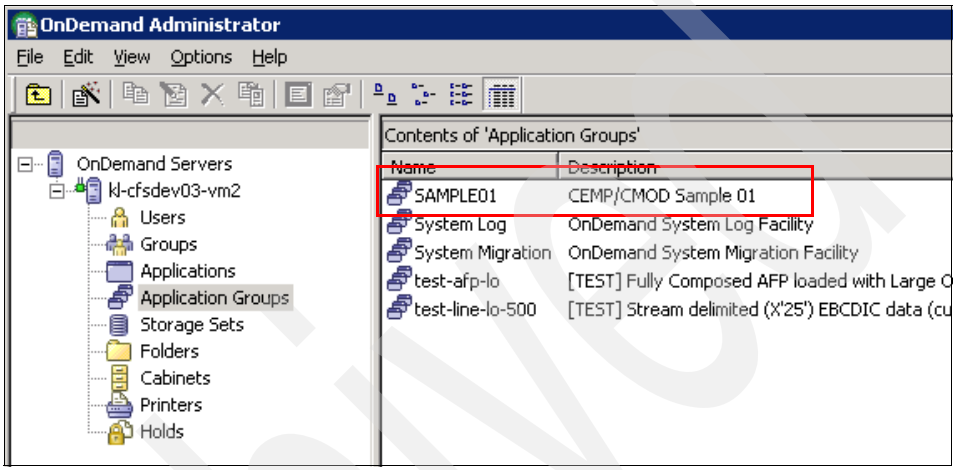



*Figure 4-15 OnDemand Administrator*

Follow these steps to create a new custom document class in the Content Engine:

1. Launch Enterprise Manager and log in as the administrator user.
2. Right-click the **Document Class** node, and select **New Class**.
3. Click **Next** on the welcome window.
4. Enter a name for the document class, and click **Next**. The name has to be unique within the object store. Type `OD-Sample01` for this example.
5. Click **Next** on the Select Properties window.
6. Click **Next** on the Select Property Attributes window.
7. Click **Next** on the Specify Content Storage Parameters window.
8. Click **Next** on the Configure Auditing window.
9. Click **Finish**.

## 4.5.2 Add properties to the new custom document class

The new custom document class that was created in 4.5.1, “Create a custom document class in the Content Engine” on page 168 does not have any custom properties. It is necessary to add new properties to this document class that correspond to the CMOD application group fields.

For example, the SAMPLE01 application group has the following fields, as shown in Figure 4-16.

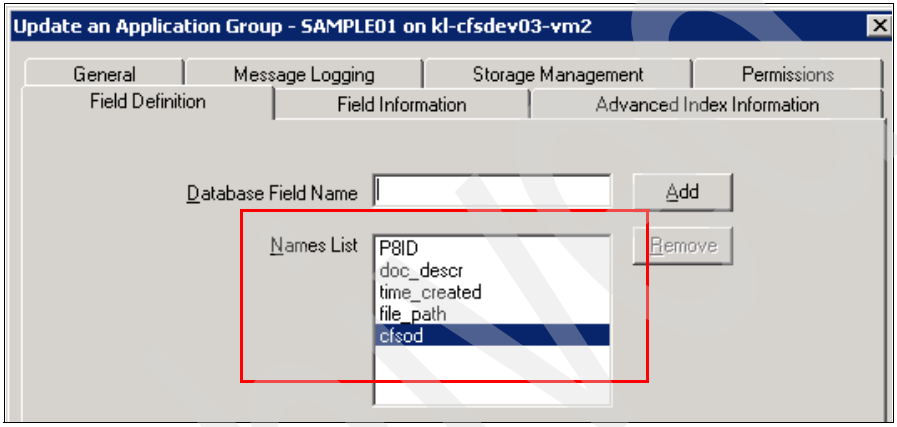



*Figure 4-16 Application group fields*

Follow these steps to add new properties to the document class that correspond to the SAMPLE01 application group fields:

1. Right-click the new custom document class (**OD-Sample01** in our example), and select **Add Properties to Class**.
2. Click **Next** on the welcome window.
3. Click **New** to launch the Create a Property Template Wizard.
4. Click **Next** on the Create a Property Template welcome window.
5. Enter a name and select an appropriate data type. The property names must be unique within the object store. Table 4-1 on page 171 shows a list of properties and their data types that need to created for our example.

*Table 4-1 Add properties to class*

| Content Engine property name | Data type |
|---|---|
| OD-P8ID | Integer |
| OD-doc_descr | String |
| OD-time_created | DateTime |
| OD-file_path | String |

6. Click **Next** on the Select a Choice List window.
7. Select **Single**, and click **Next**.
8. Click **Finish**.
9. Repeat steps 2 - 8 for each property that is shown in Table 4-1.
10. After all of the properties are created, you return to the Add Properties to a Class Wizard. Click **Next** on the Select Properties window.
11. Click **Next** on the Select Property Attributes window.
12. Click **Finish** to complete the Add Properties to Class Wizard process.

Figure 4-17 shows the properties in the Content Engine after following the previous steps.

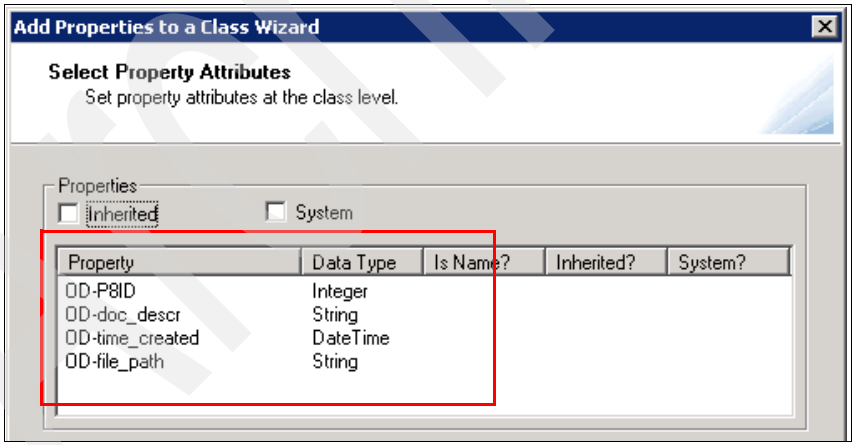



*Figure 4-17 Properties shown in Enterprise Manager*

Archived

### 4.5.3 Map Content Engine properties to CMOD application group fields

Mapping the CMOD application group fields to the Content Engine properties allows you to federate the metadata to the Content Engine. A Content Engine document class can be mapped to only one OnDemand application group. Similarly, a Content Engine property can be mapped to only one field in an application group.

Follow these steps to map the Content Engine properties to the OnDemand application group fields:

1. Launch Enterprise Manager, and log in as the administrator user.
2. Right-click the object store, and select **Properties.**
3. Select the **Content Federation** tab.
4. Click **Add/Edit**.
5. In the Fixed Content Device drop-down menu, select the CMOD fixed content device that was created in “Create the CMOD fixed content device” on page 165.
6. After a fixed content device is selected, a list of CFS-enabled application groups is displayed in the Document Classes panel. Select a CFS-enabled application group.
7. In the Content Engine document classes panel, select the custom document class that was created in 4.5.1, “Create a custom document class in the Content Engine” on page 168. For example, select the **OD-Sample01** document class.
8. After choosing an application group and a Content Engine document class, click **Add**.
9. Map the application group fields to the Content Engine properties, as shown in Figure 4-18 on page 173.

   Select an application group field from the External Repository Available Properties list, and select a matching Content Engine property from the Content Engine Available Properties list. After choosing a field and a property, click **Add**.

   Repeat this mapping for all of the properties. Click **OK** when the mapping is completed.

Archived

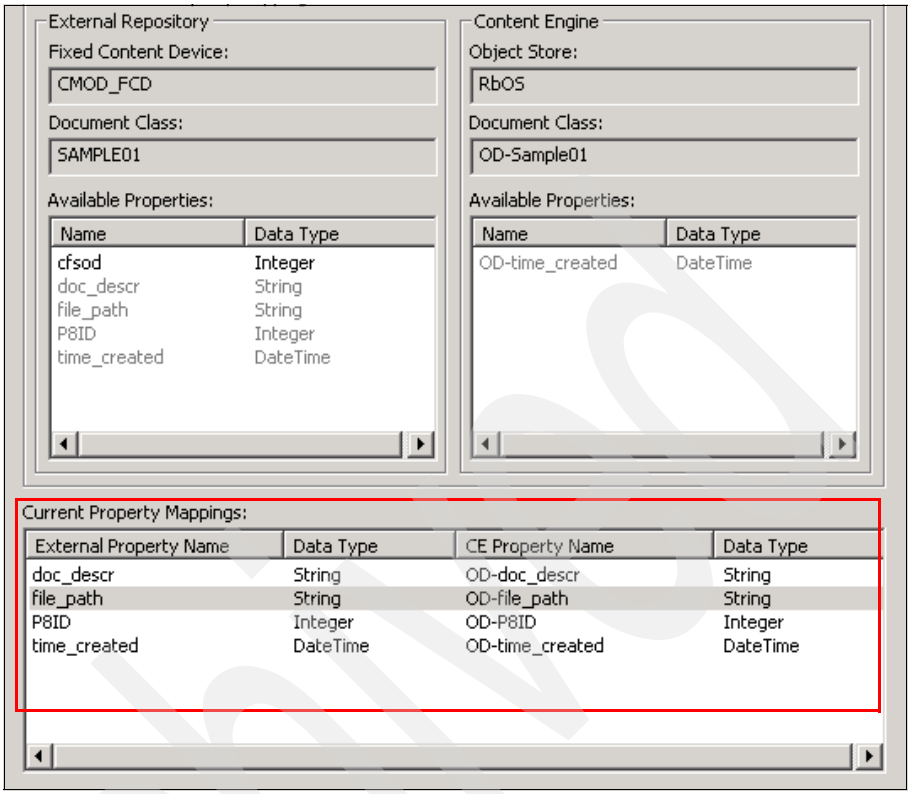



*Figure 4-18 Mapping application group fields to Content Engine properties*

**Mapping:** Map one application group field to the DocumentTitle property in the Content Engine. Although this mapping is not required, this mapping helps when querying for the federated CMOD documents in WorkplaceXT.

The cfsod field is used for maintaining the status of CFS-CMOD. There are no values added to your federated CMOD documents when mapping this field to a property in the Content Engine.

## 4.5.4 Configure and run the CFSOD exporter utility

The CFSOD exporter utility is a Java application that uses JDBC drivers to connect to the OnDemand database tables and CFS importer tables. The CFSOD exporter utility is designed to run anywhere, but we recommend that you

run it on the Content Engine. Refer to 4.3, “Planning” on page 151 for details about the benefits of running the exporter on the Content Engine.

The CFSOD exporter utility is shipped with the OnDemand Web Development Kit, as shown in Figure 4-19.

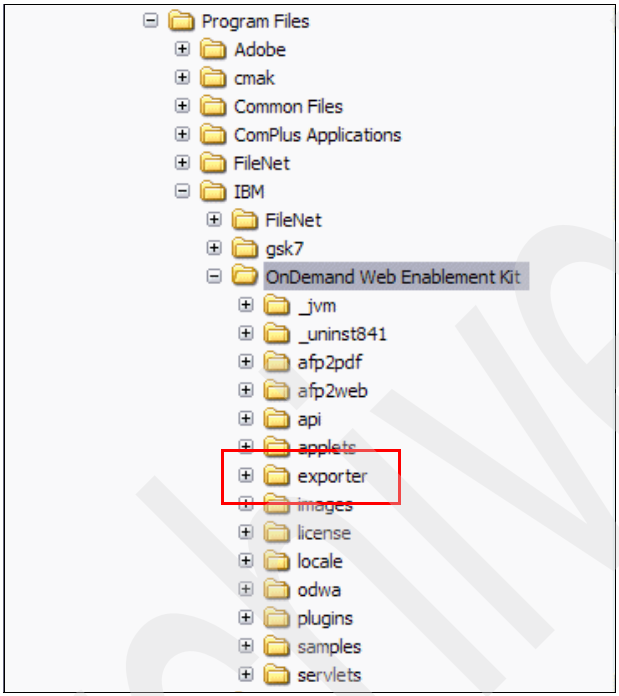



*Figure 4-19 CFSOD exporter utility in OnDemand Web Development Kit*

Typically, you use this command line execution of the CFSOD exporter utility:

```
java -classpath <jdbc driver>;CFSODExporter.jar [configure/federate]
[options]
```

The [configure] action is used to save all the [options] specified to a configuration file. Also, this option is a good utility for testing database connectivity.

The following options are supported:

- **-u** *<user ID[, user ID2]>* This user ID is needed to authenticate to the database instance.
- **-p** *<password[, password2]>* This user’s password is needed to authenticate to the database instance.

- **-o** *<port[, port2]>* This connection port number is associated with the database instance.
- **-e** *<db engine[, db engine2]>* Specify the database engine (DB2, MSSQL, or ORACLE).
- **-h** *<hostname[, hostname2]>* Specify the database server host name.
- **-I** *<db instance[, db instance2]>* Specify the database instance name.
- **-d** *<path>* Specify the directory in which trace files can be stored.
- **-t** *<export trace>* This option enables tracing for the utility. A value of 0 disables trace and a value of 1 enables trace. This value is optional.
- **-g** *<AG name>* Application Group Name to process. This value is optional.
- **-n** *<number>* Specify the number of documents to process per session. Provide a number between 1 and 250. This value is optional.
- **-D** *<number of days>* Specify how many days you want to go back from today (prior to today) to export the data. This value is optional.
- **-s** *<db owner>* Specify the OnDemand database owner (z/OS only - required if more than one OnDemand instance is defined to DB2).
- **-f** *<path>* If the configure action is used as described previously, this value is the complete path and filename to write the configuration parameters file. Otherwise, this value is the complete path and filename from which to read the configuration parameters.

**Note:** The -u, -p, -h, -o, -e, and -I options can have two comma-separated parameters:

- Parameter one always points to the OnDemand database.
- Parameter two (optional) always points to the CFS database that is defined in the Java Naming and Directory Interface (JNDI) data source for the Content Engine fixed content device.

## Generating a configuration file

You can generate a configuration file with all of the necessary information for the CFSOD exporter utility to use. Example 4-1 on page 176 is an example of generating a configuration file where the CFS importer database tables reside on the Content Engine instead of the OnDemand database server.

*Example 4-1 Generate CMOD export configuration file*

```
java -cp
"C:\jdbc\db2jcc.jar";"C:\jdbc\db2jcc_license_cu.jar";"C:\Program
Files\IBM\OnDemand Web Enablement Kit\exporter\CFSODExporter.jar"
com.ibm.cm.od.exporter.Exporter configure -u odadmin,ceadmin -p
oduser,tester -e db2,db2 -I ARCHIVE,CMODDB -h kl-cfsdev03-vm2,cm-cfsrb
-o 50000,50000 -d C:/CMODExporter/trace -n 200 -t 1 -f
C:/CMODExporter/config/exporter_fcd.config
```

When this command is run, the `exporter_fcd.config` file is generated, and it is ready for the CFSOD exporter utility to use.

### Running the CFSOD exporter utility

After a configuration file is generated, running the CFSOD exporter utility is simple. Example 4-2 is an example of running the CFSOD exporter utility using a configuration file.

*Example 4-2 Running the CMOD exporter utility*

```
java -cp
"C:\jdbc\db2jcc.jar";"C:\jdbc\db2jcc_license_cu.jar";"C:\Program
Files\IBM\OnDemand Web Enablement Kit\exporter\CFSODExporter.jar"
com.ibm.cm.od.exporter.Exporter federate -f
C:\CMODExporter\config\exporter_fcd.config
```

You only need to set the option to **`federate`** and to specify the exporter configuration file.

The CFSOD exporter utility is scheduled to run using the operating system scheduling utility. UNIX systems use cron. Windows systems use the Windows Scheduler. See 4.3, “Planning” on page 151 for details about when to run the CFSOD exporter utility.

## 4.6 View federated CMOD documents in WorkplaceXT

Additional configuration is needed to view the federated CMOD documents in WorkplaceXT. See the IBM FileNet WorkplaceXT product documentation for more information about how to configure the CMOD viewer in WorkplaceXT:

http://www.ibm.com/support/docview.wss?rs=3278&context=SSNVNV&context=SSTHRT&context=SSNVUD&context=SSS236&context=SSNW2F&uid=swg27010422&loc=en_US&cs=UTF-8&lang=en#xt

After you configure CMOD, Content Engine, and WorkplaceXT, and you successfully federate the CMOD documents to the Content Engine, you can view a federated CMOD document using WorkplaceXT:

1. Log in to WorkplaceXT, and click the **Search Mode** icon, as shown in Figure 4-20.

> **Tip:** The federated CMOD documents are filed in the Unfiled Documents folder of the Content Engine after federation. You cannot browse to the federated documents in WorkplaceXT. Instead, you must query for the documents using the WorkplaceXT search function.

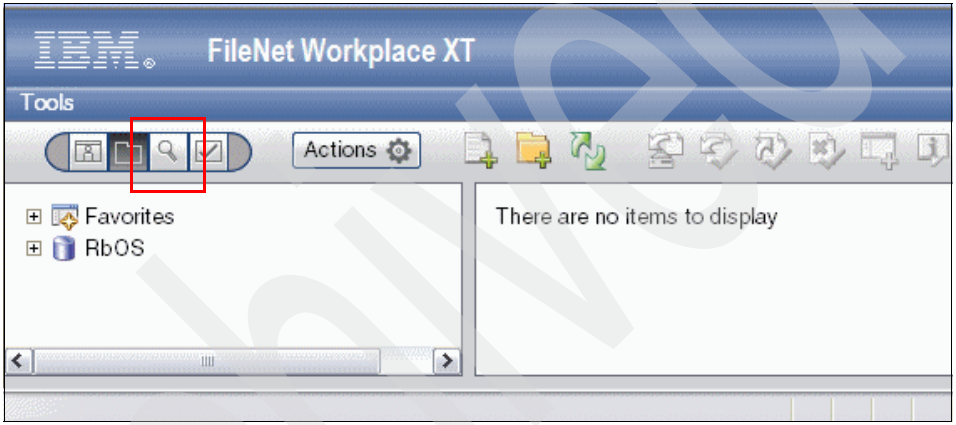



*Figure 4-20 The search mode icon in WorkplaceXT*

2. After you click the Search mode icon, the search template is displayed, as shown in Figure 4-21 on page 178. Enter the search criteria that you want, and click the **Search** button.

Archived

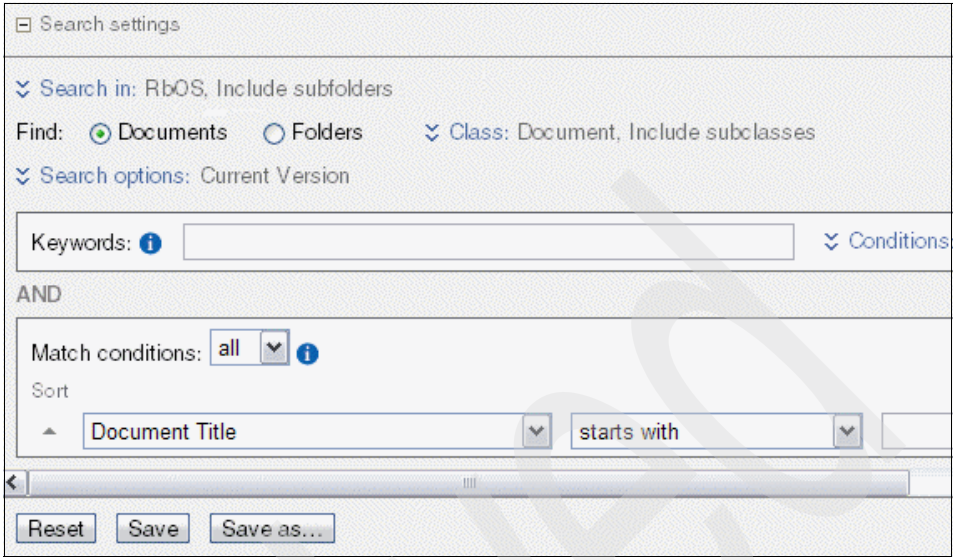



*Figure 4-21 WorkplaceXT search template*

Figure 4-22 shows the search result in WorkplaceXT. All of the federated CMOD documents have the application/vnd.ibm-OnDemand Multipurpose Internet Mail Extensions (MIME) type in the Content Engine regardless of the actual document type that is stored in the OnDemand server.

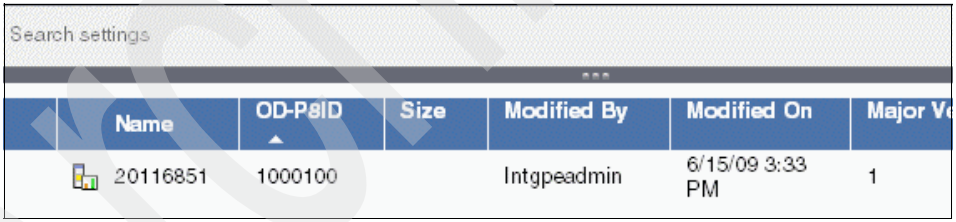



*Figure 4-22 Search results shown in WorkplaceXT*

3. You can double-click the document from the search result pane in WorkplaceXT to launch the CMOD viewer. The CMOD viewer determines the actual document type and launches the appropriate viewer. For example, if the actual document is an Advanced Function Presentation (AFP) document, it launches the AFP viewer. If it is a Line Data document, it launches the Line Data viewer.

Archived

**WorkplaceXT:** The CMOD viewer is only supported in WorkplaceXT. There is no support for viewing the federated OnDemand documents in Workplace.

## 4.7 Federated records management with CFS-CMOD

IBM Content Manager OnDemand is designed to handle an extremely high volume of document ingestion to the system, typically of a static nature, such as credit card or bank statements. Each document ingestion might contain thousands of individual documents or pages.

Content Manager OnDemand offers a retention feature, which allows you to set the document retention for a fixed period at the document ingestion time. One example is an investment company that applies a simple retention policy of eight years on all of their customer statements.

CMOD cannot apply an event-based retention policy that is based on, for example, the date that the customer closed an account. In this scenario, the clock does not begin the eight year period until the customer closed the account.

By enabling records federation services using CFS-CMOD, it becomes possible to manage Content Manager OnDemand content in a manner that is consistent with your organization's records retention policies.

When federated and declared, Content Manager OnDemand content can be tied to dynamic retention policies, such as account closure, policy termination, contract execution, or any other event. In these circumstances, records federation services can allow your organization to retain content for a certain amount of time starting on the date of the event.

Companies must take into account special design considerations to properly manage a large collection of data when the company deals with a variety of regulatory policies and litigation. See the white paper, *Compliance and Retention Management with IBM Content Manager OnDemand,* at this Web site for more information:

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101378

### 4.7.1 Records declaration

The records federation services enablement of OnDemand provides flexible options for declaring and classifying content as records by utilizing Enterprise

Records (formerly known as IBM FileNet Records Manager) to declare and manage record holds.

Records federation services provide these capabilities:

- Automatically lock down Content Manager OnDemand documents while they are ingested into the system. This function is used as a performance option.

  Enabling this option means that CMOD assumes that Enterprise Records controls expiration processing. Not setting this option means that every time that a document is placed under Enterprise Records control, P8 notifies CMOD to lock down the document. Not setting this option can result in an extremely high degree of requests coming to CMOD. Thus, by using this option, CMOD is able to avoid this overhead.
- Manually lock down individual Content Manager OnDemand documents.
- CMOD documents can be placed under Enterprise Records control after they are locked down.
- Enterprise Records can apply holds to the source Content Manager OnDemand documents so that the disposition schedule is halted until the hold is released.
- Use the rich Enterprise Records toolset to define and update the record life cycle for a document.
- Override the retention rule that is applied to a particular load, and apply record holds to individual documents.

We recommend locking down CMOD documents automatically when they are ingested into the system, then, immediately using Enterprise Records to declare these documents as records. After a document is declared as a record, Enterprise Records takes over the control of the document and administers security permissions to insure that only authorized personnel can view and access the document.

### 4.7.2 Declaring federated CMOD documents as records

There are two ways to lock down the CMOD documents. First, you can federate the CMOD documents normally to the Content Engine using CFS-CMOD and use Enterprise Records to declare the federated documents as records and lock down the source CMOD documents in the OnDemand server. Second, you can configure the application group to federate and lock down the source documents simultaneously and use Enterprise Records to declare the federated documents as records.

After a CMOD document is successfully federated, you can use WorkplaceXT to declare the document as a record manually.

**Tip:** Enterprise Records is a separate product, and it must be configured with WorkplaceXT to declare documents as records manually.

Figure 4-23 shows how the federated CMOD document is displayed after it has been declared as a record. Notice that the lock icon is displayed next to the name. The lock icon indicates that the document has been declared as a record by Enterprise Records.

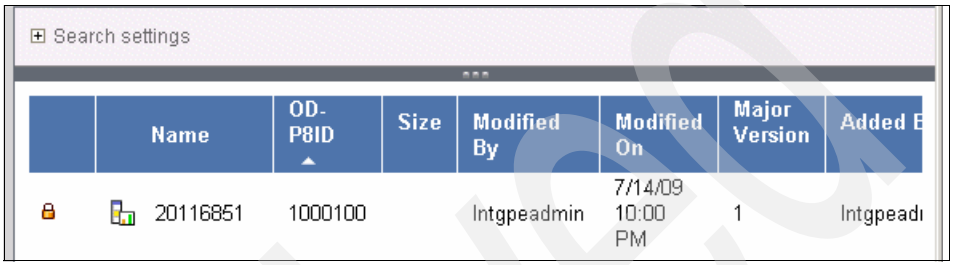



*Figure 4-23 Federated CMOD document declared as a record*

Figure 4-24 shows that the corresponding source CMOD document is locked down. The lock icon is also displayed next to the name, and the icon indicates that the document has been locked down in the OnDemand server.

*Figure 4-24 Locked down source CMOD document shown in OnDemand client application*

## 4.7.3 Lockdown behavior

The ability to declare federated CMOD documents into an IBM FileNet Content Manager object store is a new feature that is supported in Content Manager OnDemand 8.4.1.

Archived

There are two ways to initiate the record declaration and, hence, the lockdown of the federated CMOD documents:

- At any time after federation
- As part of the CMOD load and federation operation

See the *IBM FileNet Content Federation Services for Content Manager OnDemand Configuration Guide*, SC19-2711, for more information about how to configure the CFS-CMOD federation and lockdown option.

### Lockdown after federation

The source CMOD documents are federated normally during load. After they are federated, Enterprise Records is used to declare the federated CMOD documents as records, which, in turn, locks down the source CMOD documents by applying a CMOD retention hold to each source document.

### Federate and lock down during load

You can set the application group that is used for loading to federate and lock down the source CMOD documents during load. The source CMOD documents are federated normally and locked down in CMOD simultaneously. However, the federated CMOD documents are not under Enterprise Records control yet. It is necessary to use Enterprise Records to declare the federated CMOD documents as records so that the removal of the hold on the source CMOD documents is controlled by IBM FileNet Content Manager. When record declaration is called on the already locked-down source CMOD documents, no additional action is performed in CMOD.

### Holds

*Holds* can also be applied to the CMOD content. If there is a litigation event, you can easily protect content under records management with a hold, which in effect extends the protection of the document until the litigation event concludes. After documents are declared as records, OnDemand loads or individual documents within a load can be placed on hold. A hold prevents the data from being accidentally altered or deleted, or intentionally (maliciously) altered or deleted. It suspends the normal destruction or disposition of that data until the hold is released.

## 4.7.4 Deletion behavior

*Disposition* refers to the processes or actions that are taken on a record or a group of records at the end of a retention period.

At disposition time, Enterprise Records verifies that the document is eligible for disposition. To be eligible, the retention period has expired, and there must not be a hold on the document.

The deletion works slightly differently depending on whether the federated Content Manager OnDemand documents have been declared as records. When the federated Content Manager OnDemand documents are declared as records, Content Manager OnDemand places a hold on the source document and excludes the source document from the deletion process even if the source document's expiration date is reached.

The following scenarios describe the delete behaviors for the federated and source Content Manager OnDemand documents:

- Federated Content Manager OnDemand documents *have not been declared as records,* and they are ready to be deleted from IBM FileNet Content Manager:

  The federated Content Manager OnDemand documents are deleted from the Content Engine. If the expiration date on the source Content Manager OnDemand documents has not been reached yet, the source documents are not deleted until the expiration date is reached. When the expiration date is reached, the source documents are deleted.

- Federated Content Manager OnDemand documents *have not been declared as records* and exist in the Content Engine; the expiration date for the source Content Manager OnDemand documents is reached:

  The CMOD server generates a delete request for the CFS-CMOD Importer, which then deletes the federated CMOD documents from the Content Engine, which differs from other types of CFS products. In other implementations, such as a CFS implementation (using IBM Content Integrator), the delete process is initiated through the Content Engine only.

- Federated CMOD documents *have been declared as records*, and they are ready to be deleted:

  When the records are deleted from Enterprise Records, the source documents are deleted immediately, regardless of the expiration date on the source documents.

In this example, Figure 4-25 on page 184 shows that the expiration date is set to 90 days for the application group. However, when the record life cycle is managed by Enterprise Records, the document is not deleted even after the expiration date (which was set originally by Content Manager OnDemand) has been reached. Only Enterprise Records can delete the source document if the document is declared as a record.

Archived

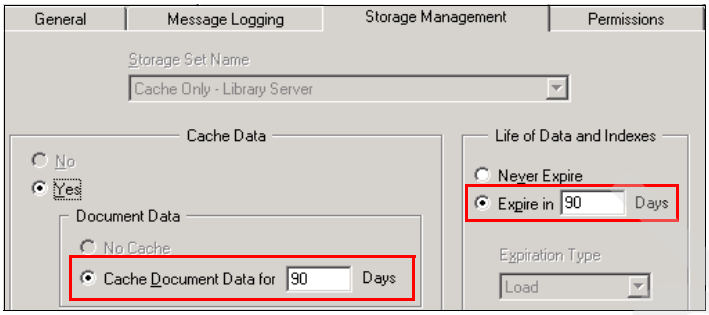



*Figure 4-25 Expiration date shown in OnDemand administration application*

When the deletion is initiated from Enterprise Records, the source document is deleted immediately, regardless of the expiration date on the source CMOD document.

## 4.8 Troubleshooting

In this section, we list the common mistakes that might occur when federating the CMOD documents. Also, we list the common error codes during the import stage of the federation. Finally, we describe how to collect the trace files to further assist in problem investigation.

### Common mistakes, limitations, and prerequisites

There are a few potential mistakes, limitations, and prerequisites that you might encounter when implementing CFS-CMOD:

- A common mistake is to create the CMOD fixed content device in FileNet Enterprise Manager but to forget to create a corresponding fixed storage area. The CMOD fixed storage area is required to import the federated CMOD documents.
- After documents of an application group have been federated, the application group must not be deleted. Deleting the application group breaks the federated content of the Content Engine documents.
- CFS-CMOD does not support CMOD documents with duplicate metadata values. Try to avoid federating the CMOD document with duplicate metadata values.
- The re-export of CMOD documents is not supported. Changes to field values on the CMOD side cannot be propagated to the federated CMOD documents in the Content Engine.

Archived

- Only WorkplaceXT supports viewing the federated CMOD documents. Workplace is not supported.
- The AFP plug-in is required on the client machine to view the federated AFP documents in WorkplaceXT. Refer to 4.6, “View federated CMOD documents in WorkplaceXT” on page 176 for more information.
- When creating an object store in FileNet P8 for CFS-CMOD, you must install the CFS-IS Extensions AddOn. Mapping properties does not work if the CFS-IS Extensions AddOn is not installed.

## Error codes

The following error codes are for import requests that are rejected. These error codes are written to the error table when federating the CMOD documents:

- `109 - Property constraint violation`

  A property of the import request contains invalid data when mapped to the P8 property.

- `113 - No Fixed Storage Area associated with the Fixed Content Device`

  There is no fixed storage area associated with the fixed content device in the import request. Federated documents cannot be created unless a fixed storage area is associated with the fixed content device.

- `114 - Content lockdown failed before federation`

  The importer was unable to lock down the content of the external document before creating the federated document.

- `116 - Invalid Object Store Id`

  The target object store does not exist. This error can also be generated if the CMOD application group cannot be mapped to an object store.

- `117 - Invalid Class Id`

  The target document class does not exist. This error can also be generated if the CMOD application group cannot be mapped to a document class.

## Collect trace logging for troubleshooting

It is important to collect the trace logging files to help the support team to determine the problems correctly and efficiently. Next, we describe the steps to enable the trace logging for CFS-CMOD.

## Enable the trace logging in the Content Engine

To enable the trace logging in the Content Engine, follow these steps:

1. Open the **Enterprise Manager Properties** dialog box.
2. Select the **Trace Logging** tab.

Archived

3. Click **Enable Logging**.
4. Select the following subsystems (see Figure 4-26). Make sure to choose full details for each subsystem:
   - CFS Import Agent
   - Fixed Contend Provider
5. Click **OK.**

The trace log file is generated within the application server installation directory.

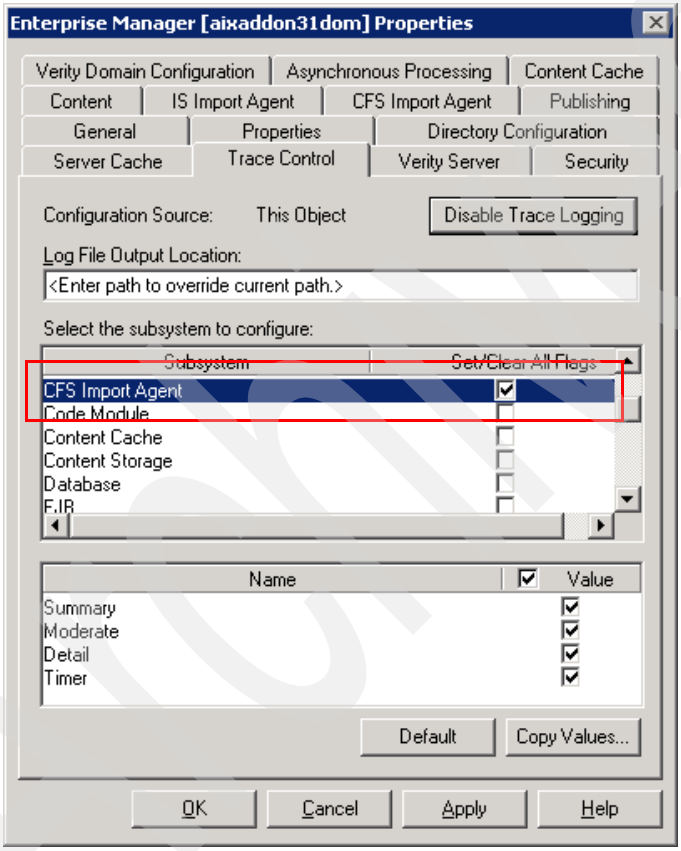



*Figure 4-26 Enterprise Manager Trace Control properties*

## Set the trace level for OnDemand Web Enablement Kit

You can set the OnDemand Web Enablement Kit trace level in the CMOD fixed content device properties page.

Follow these steps for setting the OnDemand Web Enablement Kit trace level:

1. Open the CMOD fixed contend device properties page in Enterprise Manager.
2. Set the CMOD Trace Level to `4`. See Figure 4-27.
3. Click **OK**.

The `arswww.trace` file is generated within the application server installation directory.

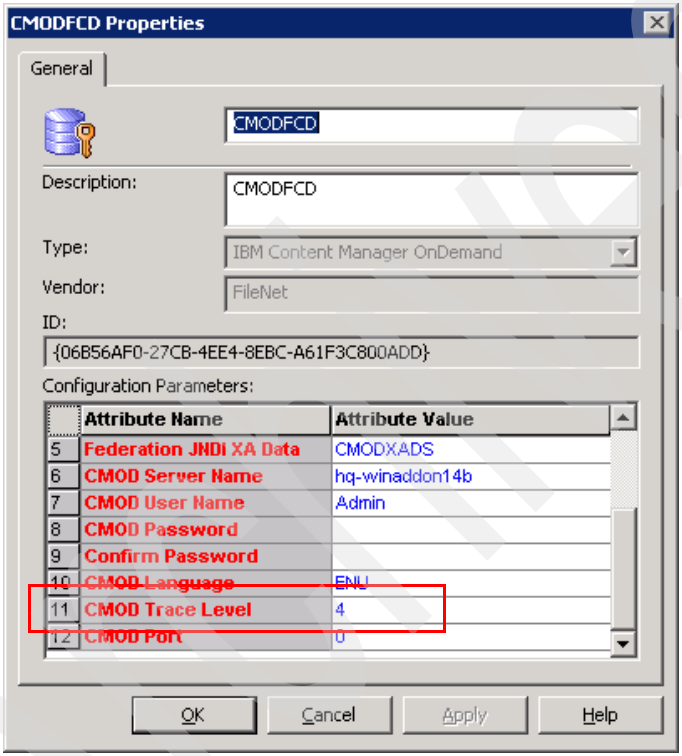



*Figure 4-27 CMOD fixed content device properties*

Archived

Archived

5

# Content Federation Services implementation using Content Integrator

IBM FileNet P8 Content Federation Services (CFS) implementation using IBM Content Integrator provides the ability to federate documents from multiple repositories, including IBM FileNet Content Manager (P8 CM), IBM Content Manager (CM8), IBM FileNet Content Services, Documentum Content Server, and Open Text Livelink Enterprise Server. Each implementation might require specific configuration. In this chapter, we discuss CFS in association with IBM FileNet Content Manager 4.5.1 and IBM Content Manager 8.4.1. We discuss how to implement CFS (using Content Integrator) with IBM Content Manager.

We cover the following topics in this chapter:

- Architecture
- Federation process
- Planning
- Installation and configuration
- Federate CM8 documents to FileNet P8
- Federated records management with CFS (using Content Integrator)
- Troubleshooting
- Best practices

© Copyright IBM Corp. 2010. All rights reserved.

# 5.1 Architecture

The CFS implementation (using Content Integrator) architecture that is described in this section expands on the CFS conceptual model that was discussed in 2.2.4, “Content Federation Services functional components” on page 41. In this section, we discuss the specific manner in which the CFS conceptual model is used for CFS implementation (using Content Integrator).

## 5.1.1 Mapping the Content Integrator data model to P8

Before discussing the architectural components of the CFS implementation (using Content Integrator), we briefly describe how the Content Integrator data model maps to the P8 data model.

The Content Integrator data model is already an abstraction of the source repository data model. So first, we describe the Content Integrator data model mapping. Then, we describe the specific CM8 connector mapping to the P8 model.

The Content Integrator data model maps to the P8 model in this manner:

- A Content Integrator *ItemClass* corresponds to a P8 *document class*.
- A Content Integrator *Content RepoItem* corresponds to a P8 *document*.
- *Properties* of an Content Integrator document correspond to *properties* of a P8 document.
- *Pages* of an Content Integrator document correspond to *content elements* of a P8 document.
- A P8 object store can contain multiple *document classes.* Hence, it can contain mappings to multiple Content Integrator Item Classes for that object store.
- An Content Integrator *ItemClass* cannot be mapped to more than one P8 object store.

The CM8 data model maps to the P8 model in this manner:

- A CM8 *Item Type* corresponds to a P8 *document class*.
- A CM8 *Item* corresponds to a P8 *document*.
- *Attributes* of a CM8 document correspond to *properties* of a P8 document.
- *Document Parts* (only *ICMBase* and *ICMTextBase* are supported by Content Integrator) of a CM8 *Item* correspond to *content elements* of a P8 document.

- A P8 object store can contain multiple *document classes*. Hence, it can contain mappings to multiple CM8 *Item Types* for that object store.
- A CM8 *Item Type* cannot be mapped to more than one P8 object store.

## 5.1.2 Components

Figure 5-1 shows the CFS implementation (using Content Integrator) components that make up the CFS conceptual model.

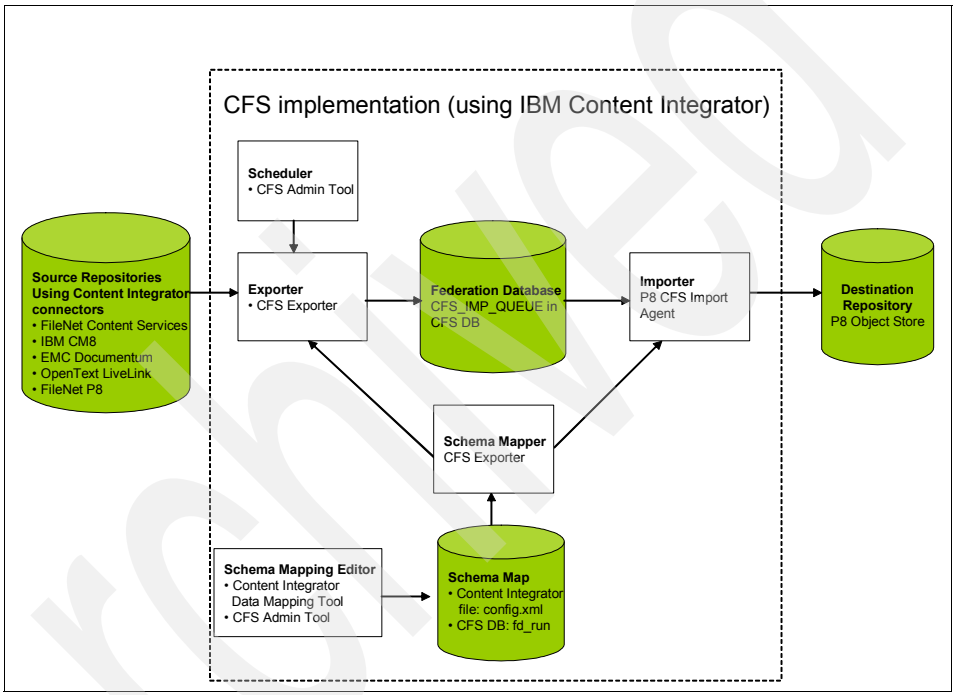



*Figure 5-1 CFS implementation (using Content Integrator) components that make up the CFS conceptual model*

The following sections describe these CFS implementation (using Content Integrator) components in more detail and discuss how they are used in the federation process.

### Schema mapping editor

To prepare Content Integrator source repository documents for federation, you create a *schema map* to map source repository properties to P8 properties. This mapping is done using the data mapping capability of the Content Integrator Administration Tool.

Archived

The results of this mapping are used by the CFS Federation Administration Tool (FedAdmin) to build the following federation rules for federating documents from a source repository to P8:

- A data map that determines how properties are translated between the source repository and the destination repository
- A query that selects documents to federate from the source repository
- A destination P8 object store and (optionally) a folder

### Schema map

The federation rules that are created using FedAdmin are stored in the fd_run CFS federation database table. The data map that is created using the Content Integrator data mapping utility is stored in the `config.xml` file in the Content Integrator home directory.

### Scheduler

FedAdmin is used to schedule when these federation rules are run.

### Source repository

CFS implementation (using Content Integrator) supports the following source repositories at this time:

- IBM FileNet Content Services
- IBM Content Manager (CM8)
- Documentum Content Server
- Open Text Livelink Enterprise Server
- IBM FileNet Content Manager (P8 CM)

Archived

## Exporter

The exporter for CFS implementation (using Content Integrator) is the CFS Exporter. This is a stand-alone application that runs continuously. It interacts with the CFS federation database and Content Integrator connectors. The exporter executes rules against an Content Integrator source repository. Documents that match the criteria are queued for federation. Rules are executed in order of priority, as indicated by the rule's location within the list of rules. See the online help topic for more information:

**FileNet P8 documentation → System Administration → Content Engine Administration → Content federation → CFS-IBM Content Integrator → The Federation Administration application → Rule Builder**

## Federation database

The federation database table that is shared by both the exporter and the importer is the CFS_IMP_QUEUE table. The CFS Exporter creates federation requests and inserts them into this table. The CFS Importer processes these requests to federate source repository documents and then removes them from this table.

In addition to using the federation database as the interface between the exporter and the importer, the federation database is used in other ways, as well. The exporter uses the federation database to store configuration information, federation queries, and scheduling information. The importer uses it to keep track of which documents have been federated and to maintain the mapping of external version series data to the FileNet P8 version series.

## Importer

The importer for CFS implementation (using Content Integrator) is the CFS Import Agent that runs as part of the P8 CM server. The importer performs the following tasks:

- Retrieves batch and content information from the federation database.
- Keeps track of what has been imported and determines if a request is a first-time import or a re-import request (property or version updates to a previously exported document).
- Creates federated documents in FileNet P8, adding the external repository reference to the content so that it can be managed by P8 applications.
- Maintains a mapping of the source repository version series data to the FileNet P8 version series in the federation database.

Archived

### Schema mapper

The schema mapper for CFS implementation (using Content Integrator) is the CFS Exporter. The Exporter uses the schema map that is created in the Content Integrator mapping tool to map properties from the source repository to FileNet P8 properties. Federation requests that are added to the federation database contain data in a format that is compatible with the P8 mapped properties.

### Destination repository

As with all CFS implementations, the destination repository is the P8 object store.

## 5.2 Federation process

The following sections describe the processing that is performed during each phase of the CFS implementation (using Content Integrator) federation process.

### Exporting

The exporting phase occurs wherever the CFS Exporter is run. When a scheduled federation rule is executed, the exporter uses Content Integrator to query the source repository that is associated with that rule to select documents to federate. It then retrieves and maps the properties from the source repository to P8 object store properties. Finally, it creates federation requests that are inserted in the CFS_IMP_QUEUE table of the federation database.

### Importing

The importing phase of CFS implementation (using Content Integrator) occurs within P8. The CFS Import Agent, if enabled, runs continuously in the P8 CM server. It periodically queries the federation database for requests inserted by the CFS Exporter. It processes document create, update, and delete requests and performs the corresponding action on the associated P8 document. If the processing completes successfully, the request is removed from the CFS_IMP_QUEUE table. If an error occurs during processing, an entry is added to the CFS_IMP_QUEUE_ERROR table. You can use CFS FedAdmin to view these errors and resubmit the requests as described in 5.6.2, “View federation rule status” on page 267.

### 5.2.1 Federated document operations

The following sections discuss the operations that are performed to manage the life cycle of federated documents.

## Create

The create operation is performed by the CFS Import Agent. The following list describes the details of this operation:

- Content Integrator source repository documents that are federated to P8 cause a corresponding P8 document to be created with the mapped properties of the source document.
- If the source repository supports versioning (such as CM8, Documentum, or Livelink), each version of the document is federated into P8 and a P8 version series object is created to represent this source document.

## Retrieve

You can retrieve the native content of Content Integrator source repository documents using any application that is written to the P8 API. It is up to the application that is retrieving this native content to understand the data format based on its Multipurpose Internet Mail Extensions (MIME) type to display or process it properly. For example, you must use WorkplaceXT when viewing federated Mixed Object Document Content Architecture (MO:DCA) documents from a CM8 repository. Only WorkplaceXT is capable of understanding the federated MO:DCA data format to display it correctly. See “Federating and viewing CM8 MO:DCA files” on page 262 for additional details.

If the source repository supports versioning, you can retrieve the content of any version of the federated document with the same restriction for MO:DCA documents described previously.

When P8 retrieves federated content, it goes back to the source repository using Content Integrator to get a stream of binary data to the native content. When P8 retrieves metadata, it returns from the P8 object store the mapped properties that were created or updated during federation.

## Update

The content of federated documents cannot change. P8 guarantees that the content of a document version does not change after it is checked in. To make any change to native content, a new version must be created. However, certain source repositories allow other behavior, so CFS must resolve these differences to insure that source document content does not change after it is checked in.

For CM8, documents whose content can change are either automatically excluded from being federated or locked down (see “Lockdown” on page 197) during federation. The particular action that is taken depends on the UsageMode setting that you have defined for your CM8 connector. If the UsageMode custom property is set to “Search and Retrieval”, these documents are excluded from federation. If the UsageMode property is set to “Records Management”, these

documents are locked down during federation. Refer to “Configuring the Usage Mode” on page 278 for additional details.

The mapped properties of a federated document version can be updated. When property updates are made on the source repository for federated documents, the following conditions apply:

- Property updates of Content Integrator source repository federated documents are propagated to P8, resulting in corresponding updates to the federated document’s properties.
- If the source repository supports versioning, property updates of a specific version of an Content Integrator source repository federated document are propagated to P8 where the corresponding version of the P8 document’s properties are updated.
- If a P8 application creates a new version of a federated document, updates from the source repository are no longer applied starting with the version that is created by the P8 application. Updates from the source repository to prior versions of the federated document are still applied.

  If a new document version is created in the source repository *after* a new version of the federated document is created in P8, that new document version is not federated. Hence, after you version a federated document using a P8 application, all future updates from the source repository are not applied, starting with that P8 document version.
- If a P8 application updates the properties of a federated document, those property updates are not sent back to the source repository.
- It is possible to update the properties of a federated document by both P8 applications and the source repository. If there is a mapping between the P8 DateLastModified property and a similar property on the source repository (for CM8, this property is LASTCHANGEDTS), this date is used to determine which properties to keep on the federated document. If the P8 federated document’s DateLastModified is newer than an update from the federated repository, the P8 document remains unchanged. Otherwise, the update request is accepted and the source repository’s properties are used to update the document.
- The Content Integrator fixed content device (FCD) has the option to set a flag to always update properties from the source repository, regardless of the P8 DateLastModified value.
- Property updates from the source repository are propagated to P8 on the next execution of the CFS exporter.

### Lockdown

To prevent updates by native source repository applications to documents that have been federated, P8 applications, such as IBM Enterprise Records (formerly known as IBM FileNet Records Manager), can initiate a *lockdown* of the document on the source repository. The following list describes the details of this operation:

- P8 only supports lockdown for source repositories that support version-specific lockdown. All repositories with the CFS implementation (using Content Integrator) support version-specific lockdown.
- When lockdown is initiated through IBM Enterprise Records, you can lock down a single version in a version series, multiple versions in a version series, or the entire version series.

### Delete

The following list describes the details of the delete operation:

- Deletes from P8 of Content Integrator federated documents are propagated to the source repository through Content Integrator.
- If the source repository supports versioning, deletes of a version of a P8 federated document are propagated to the Content Integrator source repository where the same version of the document is deleted.
- Document deletes from the source repository are not propagated to the P8 federated system, which also applies to deletes of specific versions of a document in the source repository.

  The CFS Exporter performs queries across Content Integrator source repositories, which returns results of either new or updated documents. Detecting a document deleted from the source repository requires a prohibitive amount of system resources, because it requires querying for all documents in the source repository and determining which documents that were previously federated now no longer exist. If this behavior is desired, you must design custom processes to provide this functionality.

## 5.3 Planning

There are a number of questions that you must ask when planning a CFS implementation (using Content Integrator) deployment:

- Which source repository do you want to federate?

  This answer will dictate which connector for CFS implementation (using Content Integrator) to use.

- Which content do you want to federate?

  Do you need to federate all content or only a subset?

  What is the business requirement for federation?
- How will the federated content be used?

  Are you looking to records enable the source repository content?

  Will the content be used in workflows?

  How will users access the content?
- Who are the users who will use the environment?

Several of these questions might seem elementary, but it is important to understand how the content will be used to insure a successful implementation.

Understanding which content will be used and how it will be used can help your organization choose which CFS products to use and what functionality to implement in P8 to meet business requirements.

In the following section, we outline several best practices and provide tips as to when you might want to use CFS implementation (using Content Integrator).

### 5.3.1 When to use Content Federation Services implementation (using Content Integrator)

This section discusses when you might want to use CFS implementation (using Content Integrator).

#### Compliance

As with other CFS applications, one of the most common and compelling reasons to use CFS implementation (using Content Integrator) is for regulatory compliance. By federating content, FileNet P8 can records enable documents and allow your organization to manage its life cycle using Enterprise Records. Enterprise Records can be used to dictate how long to retain content, and when and under which conditions it can be deleted. Using CFS implementation (using Content Integrator) in this manner can help your organization lower storage costs and insure regulatory compliance.

#### Single user interface

Federating content using CFS implementation (using Content Integrator) makes the source repository content visible within P8 Content Manager, which, in turn, allows your organization to provide a single point of entry for users to access content.

### Process automation/workflow

For organizations that are looking to automate their content-centric processes, federating content using CFS implementation (using Content Integrator) makes it possible for the source repository documents to actively launch and participate in workflows.

### Alternative to content migration

Perhaps your organization has looked at migrating the content from various repositories into a central location but for reasons that can range from migration costs, investments made building custom applications, user training, and many others, you have not been able to complete the migration. CFS implementation (using Content Integrator) can help. CFS implementation (using Content Integrator) can provide your organization with many of the benefits of a single enterprise repository previously outlined, without several of the inevitable hurdles that come with a physical migration.

## 5.3.2 Planning considerations

In this section, we cover several topics that can help your organization plan and deploy CFS implementation (using Content Integrator).

### Testing strategy

During the CFS implementation, organizations typically test numerous iterations of the configuration. What ends up happening is that large amounts of sample data gets ingested into the environment. Having the ability to quickly identify the content that is federated makes it easier to delete the sample data before testing a new configuration. You can use these methods to quickly identify the content that is federated:

- Search for documents that have been added by the designated CFS service user.
- Search for documents by document class, assuming that the document class is exclusive for federated content.

Keep the size of the federated document sample under 50 documents, which makes it easy to delete the sample from WorkplaceXT, because the sample documents all show up on a single page. Larger sample sets can be deleted in WorkplaceXT, but they require paging through the result pages. Alternatively, it is possible to delete large numbers of documents from the FileNet Enterprise Manager.

Be aware that after you delete a document from a P8 application, such as WorkplaceXT or Enterprise Manager, that delete gets propagated to the source

Archived

repository where it is also deleted. So, for your test environment, make a backup of your source repository before you perform any deletes from P8. Then, after you have deleted your test documents from P8 and have confirmed that those deletes have been propagated to your source repository, perform a restore on your source repository.

### CM8 documents to include in federation

Although it might be tempting to federate all CM8 content to P8, your organization must ask itself whether that is truly necessary. Federating content that will not be consumed within P8 in any way only uses up valuable system resources.

In order to determine whether content needs to be federated, your organization must ask itself the following questions. Answering yes to the questions means that the federation of the content can be beneficial:

- Will the content participate in workflows? (Requires P8 Business Process Manager)
- Does the content need to be records-enabled? (Requires Enterprise Records)
- Are users currently using P8 and having to go out of P8 to use CM8 to collect additional documents on a regular basis?
- Are you planning to replace your current CM8 user interface with P8 WorkplaceXT?

### CM8 access privileges for the CFS user

You must create a user account in CM8 for CFS implementation (using Content Integrator). CFS uses this account to select content to federate, retrieve content, set the lockdown flag, and destroy the content when instructed to do so by a P8 application, such as Enterprise Records.

The CFS user is the designated CM8 security principal that P8 Content Federation Services will use to access a specific CM8 repository. The following list identifies the minimum access privileges that CM8 must grant the CFS user on all items to be federated:

- AllowConnectToLogon
- ClientReadBasePart
- ItemDelete
- ItemQuery
- ItemSQLSelect
- ItemTypeQuery

In addition, the CFS user must also be granted the ItemRecordsAdmin privilege if CFS will be used in conjunction with Enterprise Records and if documents that are federated from CM8 will be declared as records.

### CM8 required configuration options

You must configure these options:

- Always set the new version policy for attributes to **Never create**.
- Always set the new version policy for document parts to **Always create**.
- Grant the CFS user the required minimum access rights to items that must be federated, as described in “CM8 access privileges for the CFS user” on page 200.
- Specify the correct Usage Mode: **Search and Retrieve** or **Records Management**. See 5.7.3, “Configuring and enabling Enterprise Records” on page 277.
- Enable Lockdown in the source CM8 repository if your Usage Mode is Records Management.

### CM8 best practices when used with CFS

We recommend these best practices:

- Avoid setting a maximum version limit on Item Type definitions.
- Avoid setting the Retention Period.
- Prefer Document Item types over Resource Item types.
- Avoid using Reference Attributes.
- Be aware that CM8 does not store time zone information when using DateTime properties. CM8 assumes that all of the DateTime values that it stores are in the Greenwich mean time (GMT) time zone.
- Set the appropriate UsageMode for your site.
- Set Content Versioning Policy to Always create.
- Set Metadata Versioning Policy to Never create.
- Enable CM8 Repository for Records Management if you will be using Enterprise Records.
- Avoid deleting federated documents from your source repository, except through using P8 applications.
- Map the P8 DocumentTitle property.

Archived

# 5.4 Installation and configuration

This section discusses the installation and configuration of CFS implementation (using Content Integrator). It assumes that you have the following environments installed and functioning without any CFS implementation (using Content Integrator) components:

- IBM FileNet Content Manager 4.5.1 (P8)
- IBM Content Manager 8.4.1 (CM8)

## 5.4.1 Deployment architecture

You can deploy the components of CFS implementation (using Content Integrator) among multiple host machines in various configurations. The configuration example that we discuss in the next few sections uses the deployment architecture that is depicted in Figure 5-2.

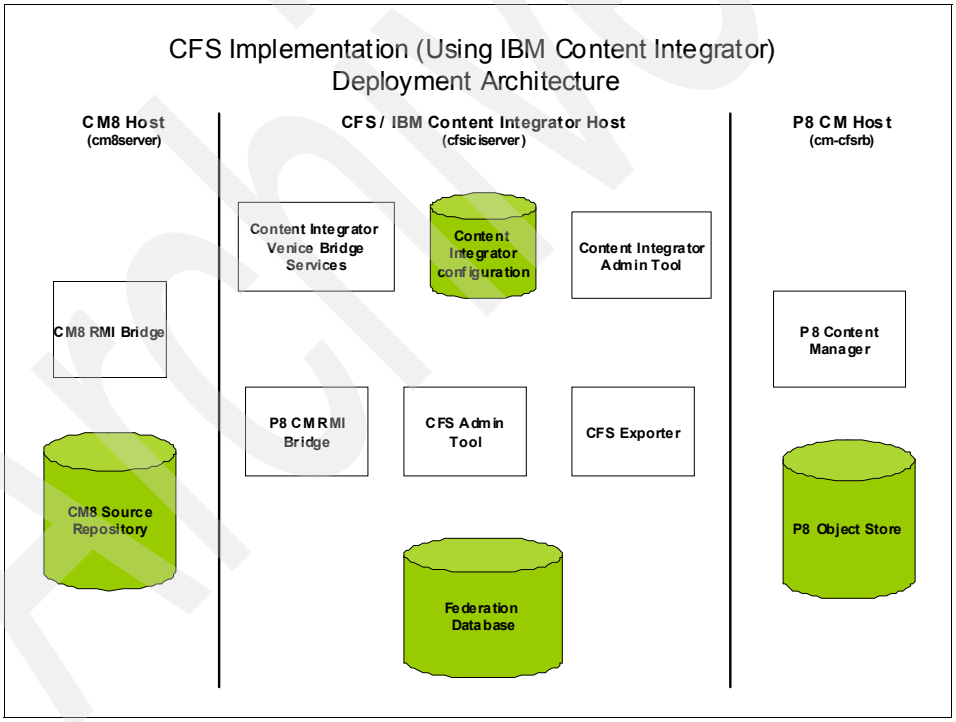



*Figure 5-2 CFS implementation (using Content Integrator) deployment architecture example*

Archived

The CFS implementation (using Content Integrator) components are divided among the Windows host machines (the host name for each machine in this example is shown in parenthesis):

- CM8 host machine (cm8server):
  - IBM Content Manager 8.4.1
  - IBM Content Integrator 8.5.1 (the installation directory is referred to as *ICI_HOME*):
    - Connector-only installation
    - CM8 Remote Method Invocation (RMI) proxy connector

      This proxy connector is the RMI proxy connector for CM8. It is started by the *ICI_HOME*\RMIBridge.bat batch file.
- CFS/Content Integrator host machine (cfsiciserver):
  - IBM Content Integrator 8.5.1 (the installation directory is referred to as *ICI_HOME*):
    - Content Integrator Venice Bridge Services

      Content Integrator Venice Bridge Services is started by the *ICI_HOME*\VeniceBridgeServices.bat file, and it provides the Content Integrator Configuration and single sign-on (SSO) services that are used by CFS implementation (using Content Integrator).
    - Content Integrator Administrator Tool

      This tool is started by the *ICI_HOME*\Admin.bat file.
    - P8 CM RMI proxy connector

      This proxy connector is the RMI proxy connector for P8. It is started by the *ICI_HOME*\RMIBridge.bat file.
    - Content Integrator Configuration

      Configuration data is stored in the *ICI_HOME*\config.xml file. This configuration data is managed by the Content Integrator Administrator Tool and read by Venice Bridge Services. Venice Bridge Services provides this configuration to RMI bridges and other Content Integrator clients, such as P8.
  - CFS 4.5.1 (the installation directory is referred to as *CFS_HOME*):
    - CFS Administration Tool (FedAdmin)

      FedAdmin is deployed in a WebSphere 6.1 application server that is located on the cfsiciserver host.

Archived

- CFS exporter

  The CFS exporter performs queries against the source repositories for documents to federate using the source repository RMI bridge, such as the CM8 RMI Bridge. The exporter extracts the metadata for these documents and creates import requests that are inserted into the federation database. These import requests are processed by the CFS Import Agents that run as part of P8. See “Importer” on page 193 for additional details.

- Federation database

  The federation database uses a DB2 9.5 instance located on the cfsiciserver host.

- P8 CM host machine (cm-cfsrb):
  - P8 Content Manager 4.5.1 is deployed in a WebSphere 6.1 application server.
  - The P8 object store is located in the DB2 9.5 database.

In the following sections, we discuss the installation and configuration of each of these components. We start with the configuration that is performed on the CM8 host (cm8server). Then, we move to the configuration that is performed on the CFS/Content Integrator host (cfsiciserver). Finally, we discuss the configuration that is performed on the P8 CM host (cm-cfsrb).

## 5.4.2 Configure security on CM8

One of your first steps in configuring your environment for CFS implementation (using Content Integrator) is to identify or create the user account in the source repository that will be used by P8 to manage federated documents in this repository.

In this example, we create a new user account in CM8 with the specific set of privileges required for CFS operations. We perform the tasks in this section on the CM8 host machine (cm8server).

### Create a CFS privilege set

We used the following steps to create a new privilege set in CM8:

1. Start the CM8 System Administration Client, and log on as an administrator user.
2. Navigate to **Authorization** → **Privilege Sets** under your CM8 library server.
3. Right-click, and select **New** → **Advanced**, as shown in Figure 5-3 on page 205.

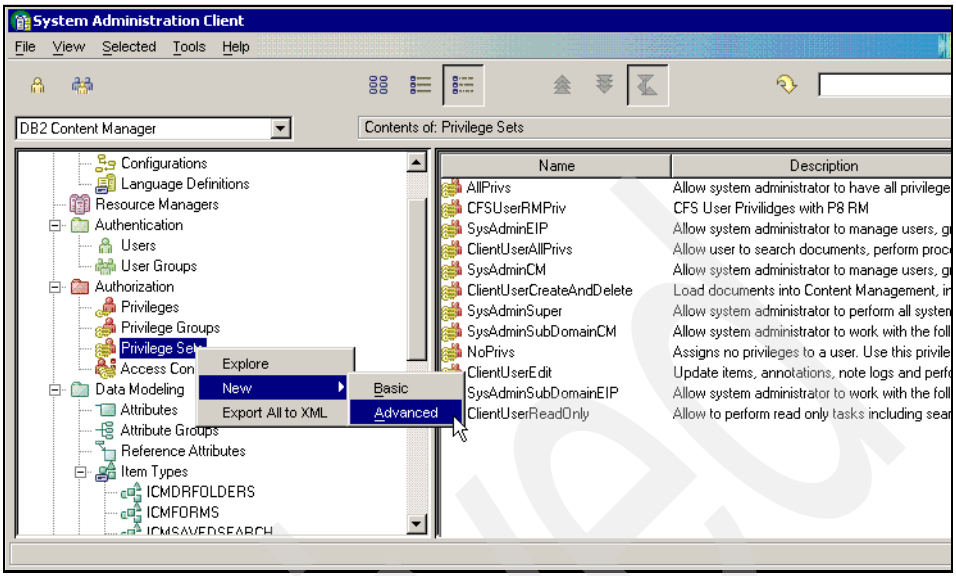



*Figure 5-3 Create a new privilege set*

4. Assign a name to this privilege set. We chose `CFSUserRMPriv`, because we plan to use this privilege set for CFS operations with Enterprise Records.
5. Select the list of privileges that are required for a CFS user, as described in “CM8 access privileges for the CFS user” on page 200.
6. When you are finished, the privilege set looks similar to Figure 5-4 on page 206.

Archived

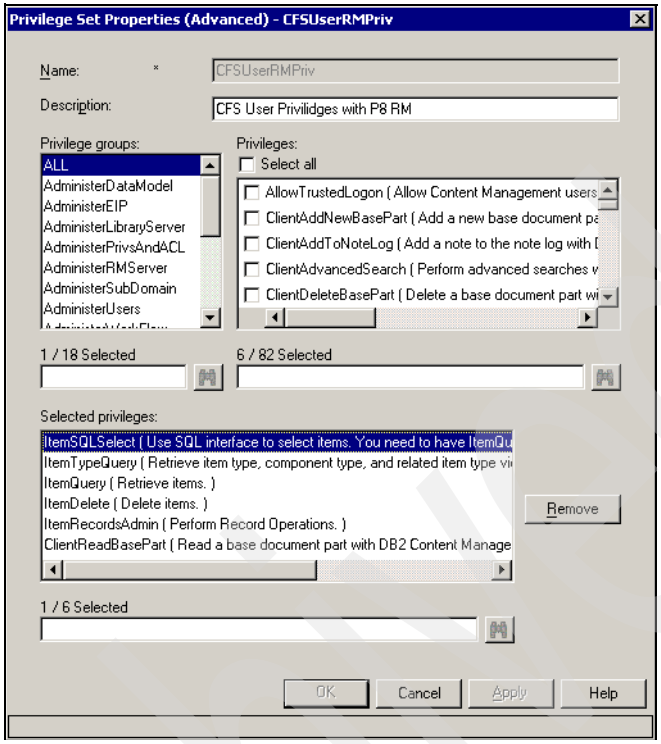



*Figure 5-4 Privilege set for CFS user*

7. Click **OK** to finish creating this privilege set.

## Create the CFS user

Next, we created the CFS user:

1. Start the CM8 System Administration Client, and log on as an administrator user.
2. Navigate to **Authentication** → **Users** under your CM8 library server.
3. Right-click, and select **New.**
4. On the New User window, we chose `CFSUSER` to be the user name of our CFS user.
5. Fill in the form, as shown in Figure 5-5 on page 207. Note that we selected the **CFSUserRMPriv** privilege set as the Maximum privilege set for this user.

Archived

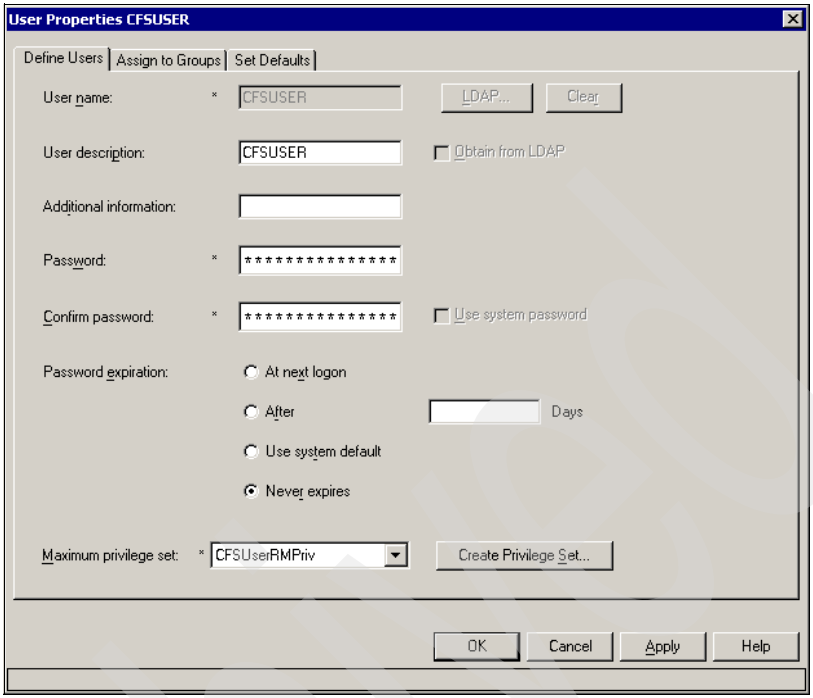



*Figure 5-5 CFS user*

6. Click **OK** to create this user.

## Modify access control lists for item types

Examine the access control lists (ACLs) for each of the item types that you plan to federate. If the existing ACLs do not provide sufficient privileges that are required by the CFS user, add the CFS user to these ACLs.

In our example, we added the CFS user to the PublicReadACL ACL using the following procedure:

1. Start the CM8 System Administration Client, and log on as an administrator user.
2. Navigate to **Authorization** → **Access Control Lists** under your CM8 library server.
3. Select the ACL that you want to modify. We chose **PublicReadACL**.
4. Right-click **PublicReadACL**, and select **Properties.**
5. In the Find groups/users section, enter `CFSUSER`, and click **Find**.
6. Select the **CFSUSER** that is found.

7. Select the **CFSUserRMPriv** privilege set in the Privilege Sets section of the form.
8. Then, click **Add**, as shown in Figure 5-6.

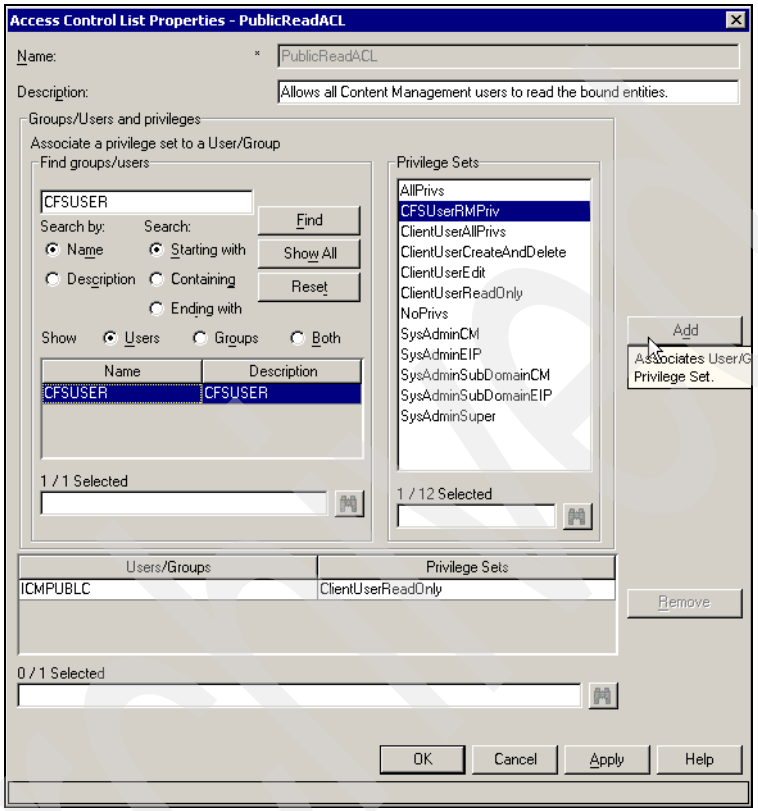



*Figure 5-6 Add CFSUSER to PublicReadACL privilege set*

9. Click **OK** to finish modifying this privilege set.

## 5.4.3 Install RMI proxy connectors

This section discusses the installation of Content Integrator and the configuration of the RMI proxy connectors as they pertain to CFS implementation (using Content Integrator). A more detailed description of the options that are available when installing and configuring each of these connectors is available in Chapter 10, “Content Integrator industry solutions” on page 499.

## Install the CM8 RMI proxy connector

As shown in Figure 5-2 on page 202, we install and configure the CM8 RMI proxy connector on the cm8server CM8 host machine. Use the following procedure to install and configure the CM8 RMI proxy connector:

1. If not already installed on the CM8 system, install the DB2 Information Integrator for Content component from the CM8 installation media. For the remaining example, the Information Integrator for Content installation directory is referred to as *II4C_HOME*.
2. Modify the `cmbicmsrvs.ini` file under the *II4C_HOME*`\cmgmt\connectors` directory. Set the values for host name, port, and database name, as shown in Example 5-1.

   *Example 5-1 Integrator for Content configuration on cm8server host*

   ```
   ICMHOSTNAME=cm8server
   ICMPORT=50000
   ICMREMOTEDB=icmnlsdb
   ICMJDBCDRIVER=com.ibm.db2.jcc.DB2Driver
   ICMJDBCURL=
   ```

3. Run the Content Integrator Installation wizard, as described in 7.1, “Installing Content Integrator” on page 322. When presented with the wizard dialog for installation type, as shown in Figure 7-4 on page 325, select **Connectors Only**. Complete the installation wizard. For the remaining example, the Content Integrator installation directory is referred to as the *ICI_HOME* directory.
4. Set the *JAVA_HOME* variable in the *ICI_HOME*`\bin\config.bat` file to the IBM Java version that is found in the WebSphere application server directory that is used by CM8, as shown:

   ```
   SET JAVA_HOME=C:\Program Files\IBM\WebSphere\AppServer\java
   ```

5. Modify the last part of the `RMIBridge.bat` file, as shown in Example 5-2.

   *Example 5-2 CM8 RMI bridge batch file configuration*

   ```
   REM The rmi registry port
   set VBR_RMIPORT=1251

   set II4C_HOME=C:/Progra~1/IBM/db2cmv8
   set CM_JARS=%II4C_HOME%/lib/cmbsdk81.jar
   set CM_JARS=%II4C_HOME%/lib/cmbview81.jar;%CM_JARS%
   set CM_JARS=%II4C_HOME%/lib/cmb81.jar;%CM_JARS%
   set CM_JARS=%II4C_HOME%/lib/db2jcc.jar;%CM_JARS%
   set CM_JARS=%II4C_HOME%/lib/db2jcc_license_cu.jar;%CM_JARS%
   set CM_JARS=%II4C_HOME%/lib/db2jcc_license_cisuz.jar;%CM_JARS%
   ```

Archived

```
set CM_JARS=%II4C_HOME%/cmgmt;%CM_JARS%

set VBR_ALLJARS=%VBR_ALLJARS%;%CM_JARS%

REM Launch the RMI Connector external VM service
java ^
-ms256m -mx256m ^
-classpath "%VBR_ALLJARS%" ^
-Dvbr.home="%VBR_HOME%" ^
-Dvbr.services.startregistry=true ^
-Dvbr.log.info=RMI_%VBR_RMIPORT% ^
-Dvbr.services.registryport=%VBR_RMIPORT% ^
-Dvbr.ejb.bridge.rmibridge.rmiurl=rmi://localhost:%VBR_RMIPORT%/RMIB
ridgeServer ^
com.venetica.vbr.ejb.bridge.rmibridge.RMIBridgeFactoryLauncher
@echo off
pause
```

These notes refer to the preceding example:

- The VBR_RMIPORT number must be unique on this host machine. If you run more than one RMI bridge on this host, each RMI bridge must have a separate port number.
- If you get a "`command line too long`" error, add the following lines after the *II4C_HOME* setting:

```
set II4C_HOME=C:/Progra~1/IBM/db2cmv8
subst x: "%II4C_HOME%"
set II4C_HOME=x:
```

## Install the P8 CM RMI proxy connector

As shown in Figure 5-2 on page 202, we install and configure the P8 CM RMI proxy connector on the cfsiciserver CFS/Content Integrator host machine. The P8 connector is used when configuring the data mapping between the source repository, such as CM8, and the target repository, such as P8.

In our example, the P8 CM RMI connector runs on the same host as the Content Integrator full installation. So, first run the Content Integrator Installation wizard, as described in 7.1, "Installing Content Integrator" on page 322. When presented with the wizard window for installation type, as shown in Figure 7-4 on page 325, select **Full**. Complete the installation wizard. For the remaining example, the Content Integrator installation directory is referred to as *ICI_HOME* directory.

The instructions for configuring a P8 CM RMI proxy connector vary slightly depending on the type of application server that P8 uses. The primary example

assumes that P8 uses a WebSphere application server. However, we also show the configuration that is required if P8 runs on a WebLogic or JBoss application server.

### *WebSphere*

The following instructions are an abbreviated form of the instructions that are described in 7.3.5, “IBM FileNet Content Manager connector configuration” on page 361:

1. Install IBM WebSphere Application Client. We refer to the installation directory as the *WAC_HOME* directory for the remainder of this example.
2. Edit the `sas.client.props` files under the *WAC_HOME*/`properties` directory. We enter the connection information to the P8 WebSphere application server:
   a. com.ibm.CORBA.securityServerHost=cm-cfsrb
   b. com.ibm.CORBA.securityServerPort=2809
   c. com.ibm.CORBA.loginSource=none
3. Copy the following files from your P8 environment to the *ICI_HOME*`\lib` directory:
   – `Jace.jar`
   – `javaapi.jar`
4. Set the *JAVA_HOME* variable in the *ICI_HOME*`\bin\config.bat` file to a Java 1.5 installation, such as the IBM Java version that is found in the WebSphere Application Client:

   ```
   SET JAVA_HOME=C:\Program Files\IBM\WebSphere\AppClient\java
   ```

5. Modify the last part of the *ICI_HOME*`\bin\RMIBridge.bat` file, as shown in Example 5-3. You can obtain details for each of these modifications in 7.3.5, “IBM FileNet Content Manager connector configuration” on page 361.

*Example 5-3 WebSphere P8 CM RMI bridge batch file modifications*

```
REM The rmi registry port
set VBR_RMIPORT=1251

set WAC_HOME=D:\Progra~1\IBM\WebSphere\AppClient
set JAVA_HOME=%WAC_HOME%\java
set PATH=%JAVA_HOME%\bin;%PATH%
set JNDI_CLIENT_PROVIDER=iiop://cm-cfsrb:2809
set
JNDI_CLIENT_FACTORY=com.ibm.websphere.naming.WsnInitialContextFactor
y

REM Launch the RMI Connector external VM service
```

```
java ^
-ms256m -mx256m ^
-Djava.ext.dirs=%WAC_HOME%\plugins;%WAC_HOME%\java\jre\lib\ext;%WAC_
HOME%\java\jre\lib;%WAC_HOME%\classes;%WAC_HOME%\lib;%WAC_HOME%\lib\
ext ^
-Djava.naming.factory.initial=%JNDI_CLIENT_FACTORY% ^
-Djava.naming.provider.url=%JNDI_CLIENT_PROVIDER% ^
-Dcom.ibm.CORBA.ConfigURL=file:/%WAC_HOME%\properties\sas.client.pro
ps ^
-classpath "%VBR_ALLJARS%" ^
-Dvbr.home="%VBR_HOME%" ^
-Dvbr.services.startregistry=true ^
-Dvbr.log.info=RMI_%VBR_RMIPORT% ^
-Dvbr.services.registryport=%VBR_RMIPORT% ^
-Dvbr.ejb.bridge.rmibridge.rmiurl=rmi://localhost:%VBR_RMIPORT%/RMIB
ridgeServer ^
com.venetica.vbr.ejb.bridge.rmibridge.RMIBridgeFactoryLauncher
@echo off
```

These notes refer to the preceding example:

- The VBR_RMIPORT number must be unique on this host machine. If you run more than one RMI bridge on this host, each RMI bridge must have a separate port number.
- When P8 CM is running in WebSphere, you *must* use the version of Java that is installed with the WebSphere Application Client. Other versions of Java will not work.
- The JNDI_CLIENT_PROVIDER must match the host name and port of your P8 CM server.
- We do *not* recommend copying the `WcmApiConfig.properties` file from P8 to the *`ICI_HOME`*`\bin` directory, as instructed in 7.3.5, “IBM FileNet Content Manager connector configuration” on page 361.

### *WebLogic*

If P8 runs on WebLogic, use the following changes to the preceding WebSphere procedure:

- Instead of installing WebSphere Application Client, copy the `weblogic.jar` file from the P8 weblogic application server directory to a directory on the local machine, such as the `C:\weblogic` directory.
- Copy the following files from the P8 environment to the *`ICI_HOME`*`\lib` directory:
  - `Jace.jar`
  - `javaapi.jar`

- Install Java 1.5, and set the *JAVA_HOME* variable in the *ICI_HOME*\bin\config.bat file.
- Modify the last part of the *ICI_HOME*\bin\RMIBridge.bat file, as shown in Example 5-4.

*Example 5-4 WebLogic P8 CM RMI bridge batch file modifications*

```
REM The rmi registry port
set VBR_RMIPORT=1251

set WL_HOME=C:\weblogic
set JNDI_CLIENT_PROVIDER=t3://cm-cfsrb:7001
set JNDI_CLIENT_FACTORY=weblogic.jndi.T3InitialContextFactory

REM add weblogic.jar to the front of the VBR classpath
set VBR_ALLJARS=%WL_HOME%\weblogic.jar;%VBR_ALLJARS%

REM Launch the RMI Connector external VM service
java ^
-ms256m -mx256m ^
-Djava.naming.factory.initial=%JNDI_CLIENT_FACTORY% ^
-Djava.naming.provider.url=%JNDI_CLIENT_PROVIDER% ^
-classpath "%VBR_ALLJARS%" ^
-Dvbr.home="%VBR_HOME%" ^
-Dvbr.services.startregistry=true ^
-Dvbr.log.info=RMI_%VBR_RMIPORT% ^
-Dvbr.services.registryport=%VBR_RMIPORT% ^
-Dvbr.ejb.bridge.rmibridge.rmiurl=rmi://localhost:%VBR_RMIPORT%/RMIB
ridgeServer ^
com.venetica.vbr.ejb.bridge.rmibridge.RMIBridgeFactoryLauncher
@echo off
```

Archived

### *JBoss*

If P8 runs on JBoss, use the following changes to the previous WebSphere procedure:

- Instead of installing WebSphere Application Client, copy the `jbossall-client.jar` file and the `log4j.jar` file from the P8 JBoss application server directory, such as the `C:\jboss-4.0.5.GA\client` directory, to a directory on the local machine, such as the `C:\jboss` directory.

  **Note:** For JBoss 5.1.0, you must copy the `jbossall-client.jar` file, as well as all other jar files that are located in the `jboss client` directory to a directory on your local machine. These additional jar files are referenced in the `jbossall-client.jar Manifest.mf` file.

- Copy the following files from the P8 environment to the *`ICI_HOME`*`\lib` directory:
  - `Jace.jar`
  - `javaapi.jar`
- Install Java 1.5, and set the *`JAVA_HOME`* variable in the *`ICI_HOME`*`\bin\config.bat` file.
- Modify the last part of the *`ICI_HOME`*`\bin\RMIBridge.bat` file, as shown in Example 5-5.

*Example 5-5 JBoss P8 CM RMI bridge batch file modifications*

```
REM The rmi registry port
set VBR_RMIPORT=1251

set JB_HOME=C:\jboss
set JNDI_CLIENT_PROVIDER=jnp://cm-cfsrb:1099
set JNDI_CLIENT_FACTORY=org.jnp.interfaces.NamingContextFactory

REM add JBoss client jar files to the front of the VBR classpath
set
VBR_ALLJARS=%JB_HOME%\jbossall-client.jar;%JB_HOME%\log4j.jar;%VBR_A
LLJARS%

REM Launch the RMI Connector external VM service
java ^
-ms256m -mx256m ^
-Djava.naming.factory.initial=%JNDI_CLIENT_FACTORY% ^
-Djava.naming.provider.url=%JNDI_CLIENT_PROVIDER% ^
-Djava.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces ^
-classpath "%VBR_ALLJARS%" ^
```

```
-Dvbr.home="%VBR_HOME%" ^
-Dvbr.services.startregistry=true ^
-Dvbr.log.info=RMI_%VBR_RMIPORT% ^
-Dvbr.services.registryport=%VBR_RMIPORT% ^
-Dvbr.ejb.bridge.rmibridge.rmiurl=rmi://localhost:%VBR_RMIPORT%/RMIB
ridgeServer ^
com.venetica.vbr.ejb.bridge.rmibridge.RMIBridgeFactoryLauncher
@echo off
```

## 5.4.4 Configure Connectors in Content Integrator Administration Tool

Now that both the CM8 RMI proxy connector and the P8 RMI proxy connector are running, we configure them in the Content Integrator Administration Tool by executing the *ICI_HOME*\bin\Admin.bat file. Use the following procedures to configure these connectors.

### Create the P8 CM connector

Use the following procedure to create the configuration for the P8 CM connector:

1. Select the **Connectors** node in the Content Integrator Administration Tool, right-click, and select **New FileNet P8 Content Manager Connector**, as shown in Figure 5-7 on page 216.

Archived

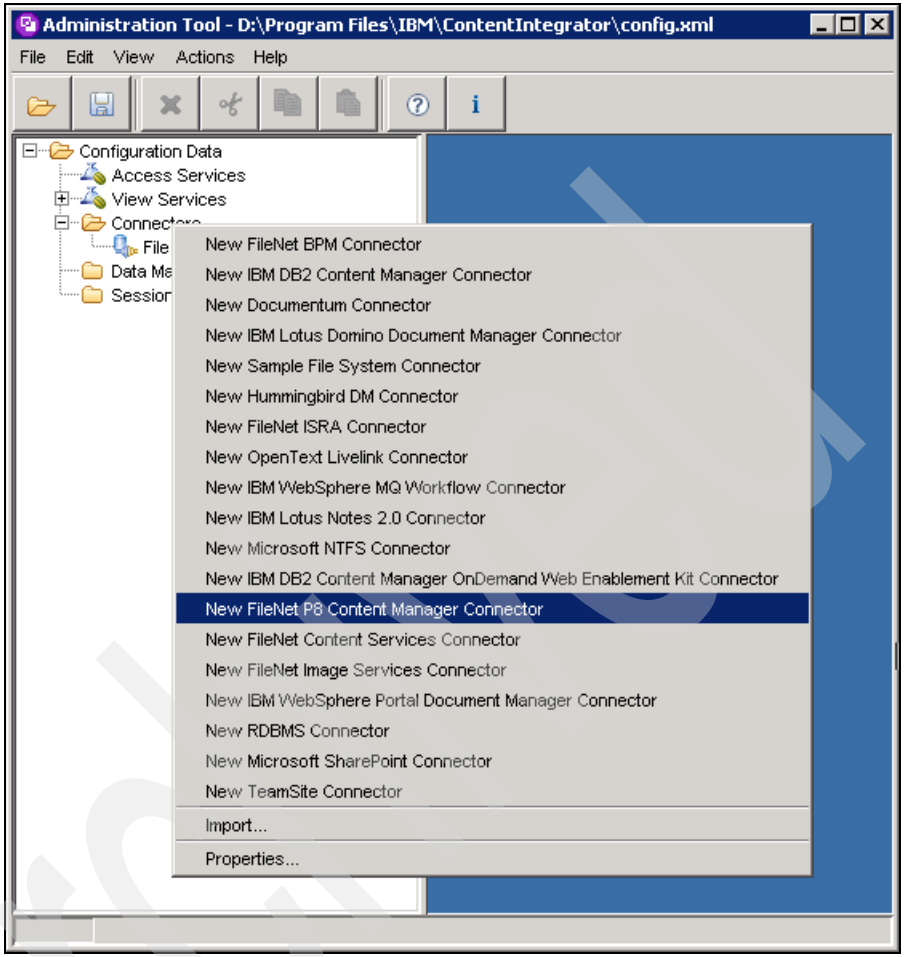



*Figure 5-7 Create new FileNet P8 Content Manager Connector*

2. On the Properties Editor window, set the fields as shown in Figure 5-8 on page 217. For a description of each of these fields, see “Connector properties” on page 337.

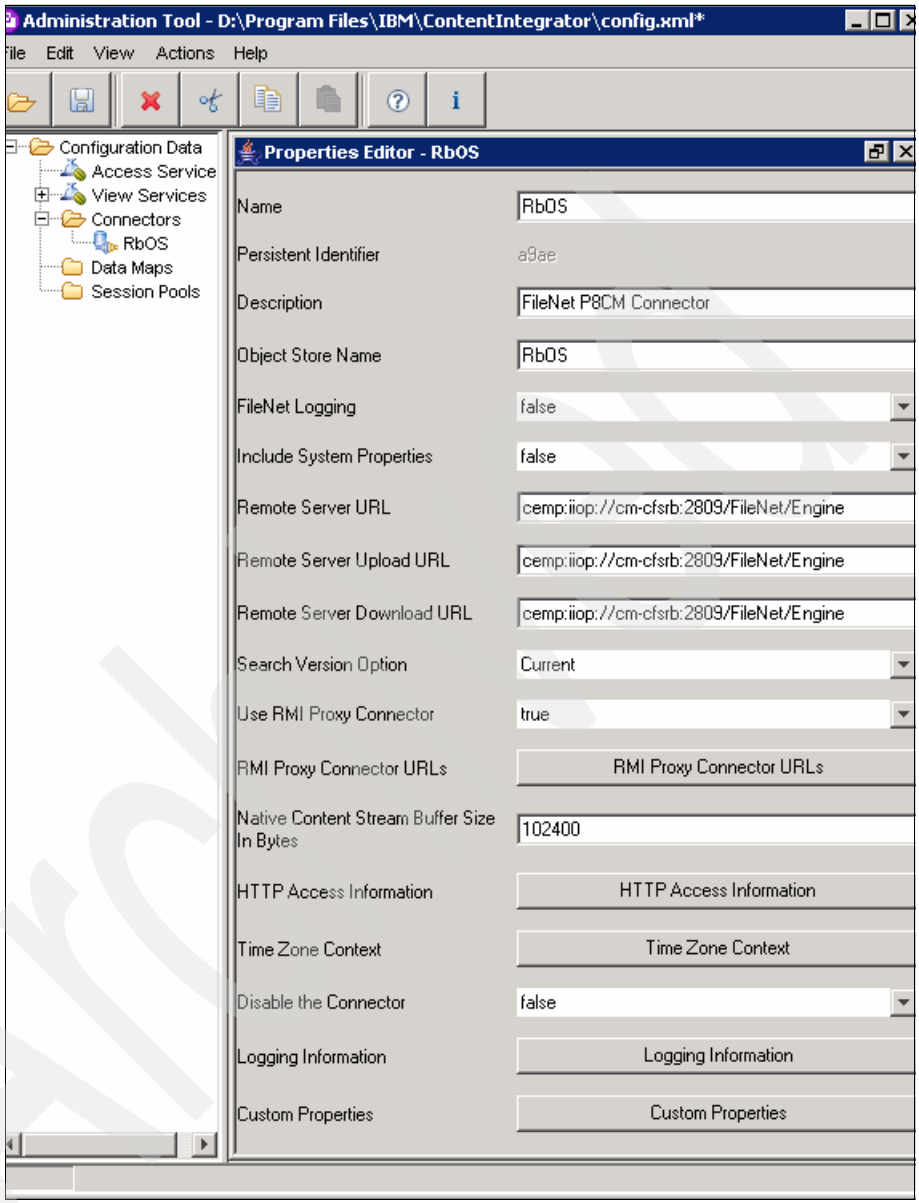



*Figure 5-8 P8 connector properties*

3. On the Properties page, you must change the fields that are shown in Table 5-1 on page 218.

*Table 5-1 P8 connector properties*

| Parameter | Comments |
|---|---|
| Name | The name must be the same name as P8 object store, for example: `RbOS`. |
| Object store name | The object store must be the P8 object store name, for example: `RbOS`. |
| Remote server URL | Enter the URL to the P8 CM server. The URL format depends on the type of application server and port that P8 CM uses, for example:<br><br>WebSphere: cemp:iiop://cm-cfsrb:2809/FileNet/Engine<br><br>WebLogic: cemp:t3://cm-cfsrb:7001/FileNet/Engine<br><br>JBoss: cemp:jnp://cm-cfsrb:1099/FileNet/Engine |
| Remote server upload URL | Use the same value as the Remote Server URL. |
| Remote server download URL | Use the same value as the Remote Server URL. |
| Use RMI proxy connector | Select **true**. |

4. Click **RMI Proxy Connector URLs**.
5. Click **Add**, and enter the host name and the port for the P8 RMI Proxy connector that you set up in “Install the P8 CM RMI proxy connector” on page 210, as shown in Figure 5-9 on page 219.

Archived

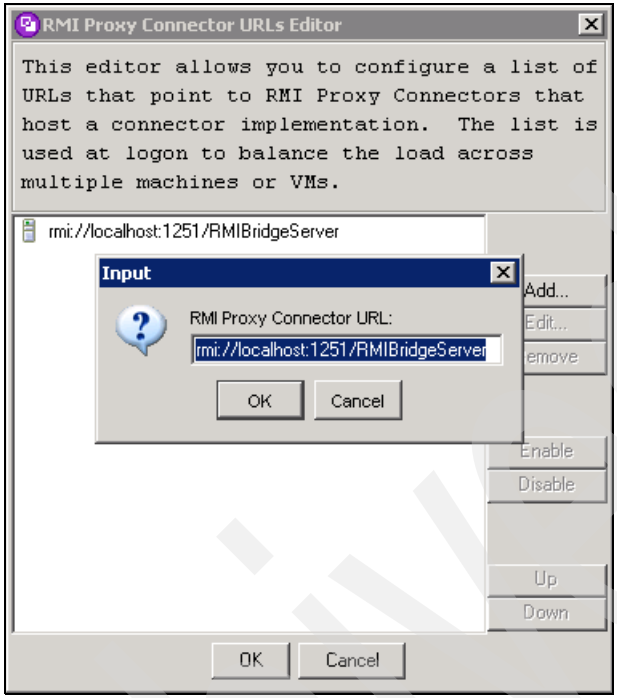



*Figure 5-9 P8 CM RMI proxy connector URL*

6. Save your configuration by selecting **File** → **Save Configuration**.
7. Make sure that your P8 CM RMI proxy connector (the *ICI_HOME*\bin\RMIBridge.bat file on cfsiciserver) is running.
8. Test the P8 CM connector by selecting it on the navigation pane, right-click, and select **Test Connection**.
9. Enter the user and password for an account that is valid on the P8 CM server.
10. Click **OK**. If successful, you get a connection succeeded message.

## Create the CM8 connector

Use the following procedure to create the CM8 connector:

1. Select the **Connectors** node in the Content Integrator Administration Tool, right-click and select **New IBM DB2 Content Manager Connector.**
2. On the Properties Editor window, set the fields, as shown in Figure 5-10 on page 220. For a description of each of these fields, see 7.3.3, “IBM Content Manager connector configuration” on page 343.

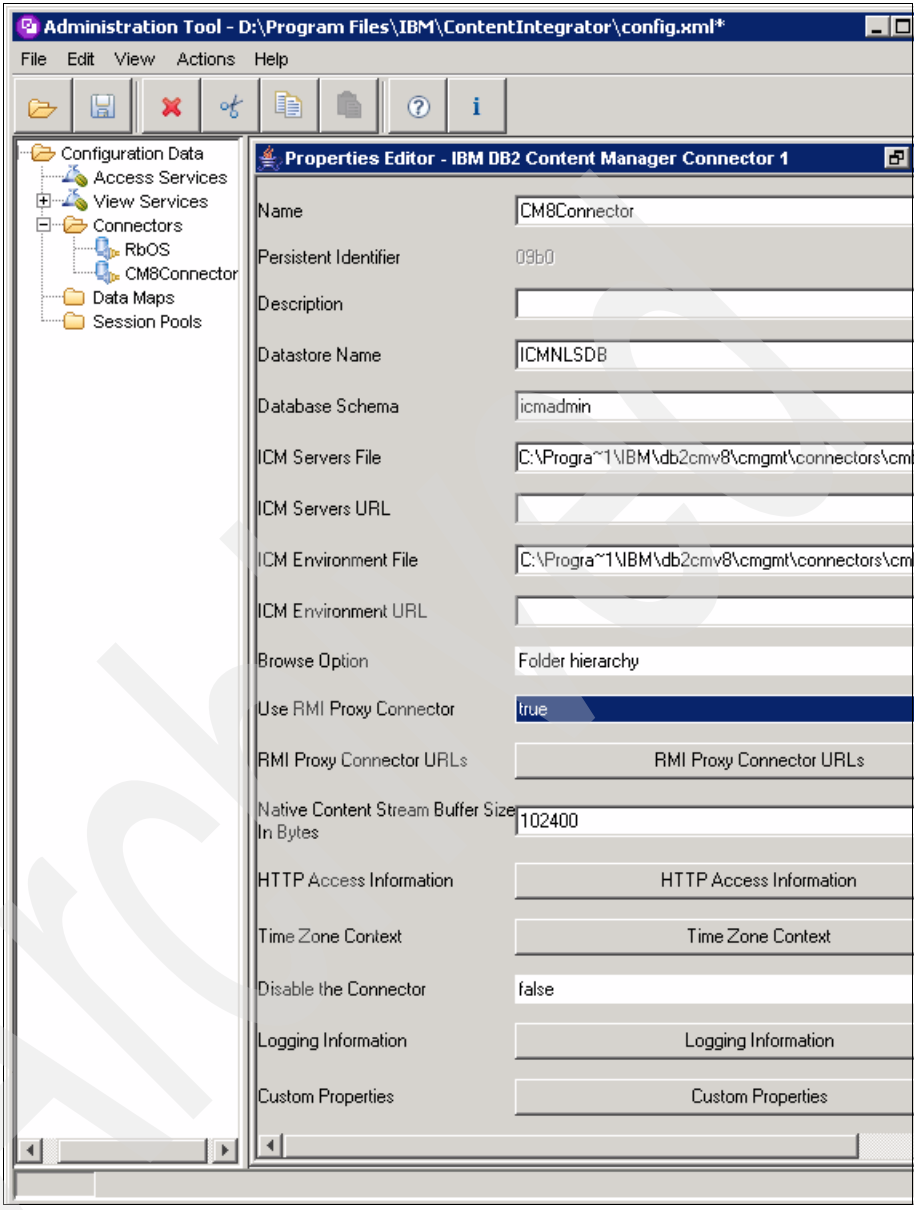



*Figure 5-10 CM8 connector properties*

3. Set Use RMI Proxy Connector to **true**. Then, click **RMI Proxy Connector URLs**.

4. Click **Add**, and enter the host name and port for the CM8 RMI Proxy connector that you set up in “Install the CM8 RMI proxy connector” on page 209 as shown in Figure 5-11.

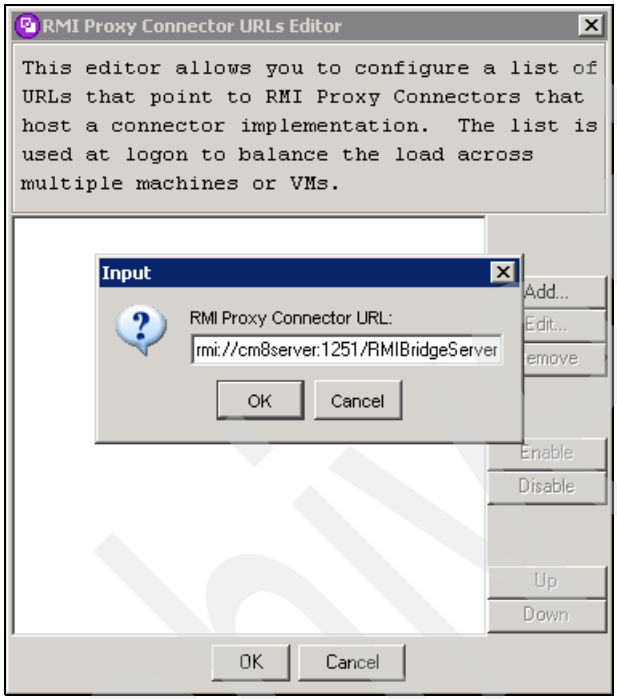



*Figure 5-11 CM8 RMI proxy connector URL*

5. Click **Custom Properties** on the Properties Editor window.
6. Click **Add** on the Custom Properties Editor window.
7. Because we are configuring this connector for Enterprise Records, we add the UsageMode property by typing `UsageMode` with the value `Records Management`, as shown in Figure 5-12 on page 222. If we were not configuring this connector for Enterprise Records, we set this value to “Search and Retrieve”.

Archived

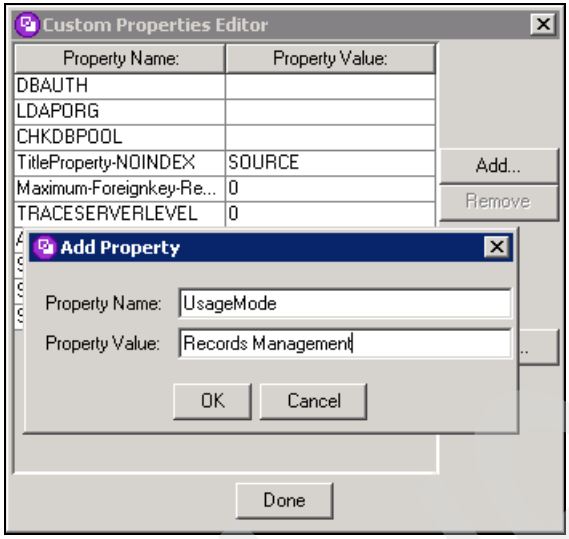



*Figure 5-12 UsageMode custom property*

8. Save your configuration by selecting **File** → **Save Configuration**.
9. Make sure that your CM8 RMI proxy connector (the `ICI_HOME\bin\RMIBridge.bat` file on cm8server) is running.
10. Test the CM8 connector by selecting it on the navigation pane, right-clicking, and selecting **Test Connection**.
11. Enter the user and password for an account that is valid on the CM8 server.
12. Click **OK**. If successful, you get a connection succeeded message.

## Create connector session pools

You must create a session pool for each connector that is used for CFS. The session pool name must match the pattern: SessionPool-*connector.* Use the following procedure to create the session pools:

1. Select the **Session Pools** node on the Content Integrator Administration Tool.
2. Right-click, and select **New Session Pool**.
3. On the session pool property panel, enter a name that corresponds to your P8 CM connector, as shown in Figure 5-13 on page 223.

Archived

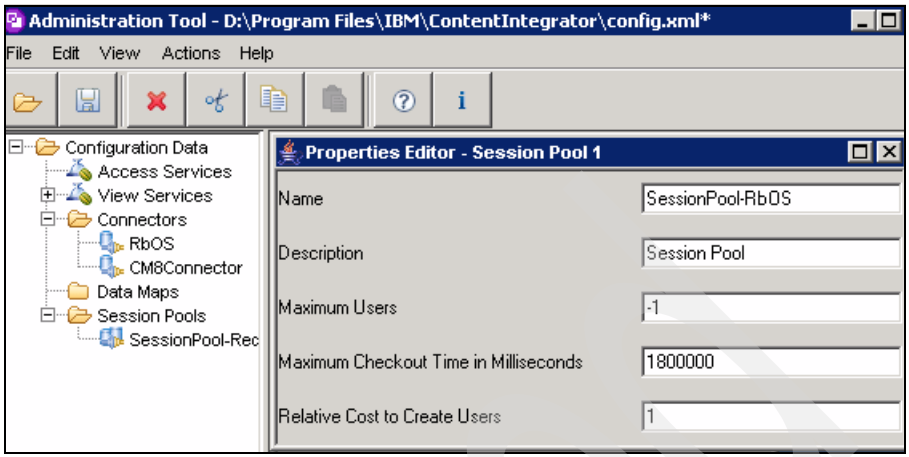



*Figure 5-13 RbOS session pool properties editor*

4. Next, select the **Session Pool Configuration** dialog connector, right-click, and select **Add Connector**. Then, select **RbOS**.

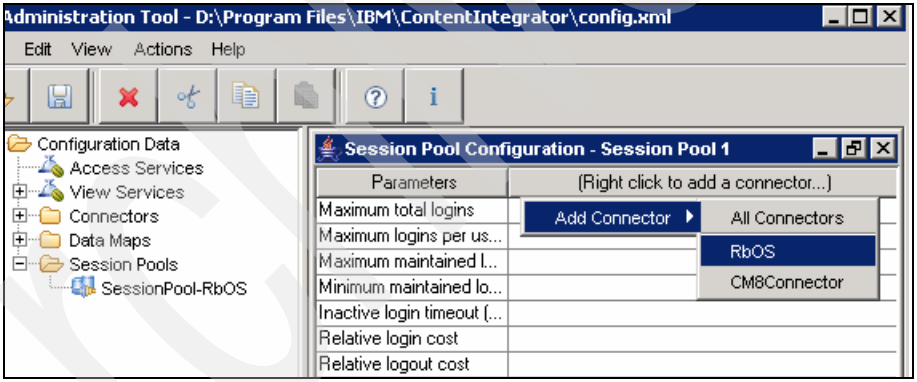



*Figure 5-14 RbOS session pool configuration*

5. Repeat steps 1-4 for your CM8 connector.
6. When you are done, your session pools look similar to Figure 5-15 on page 224.

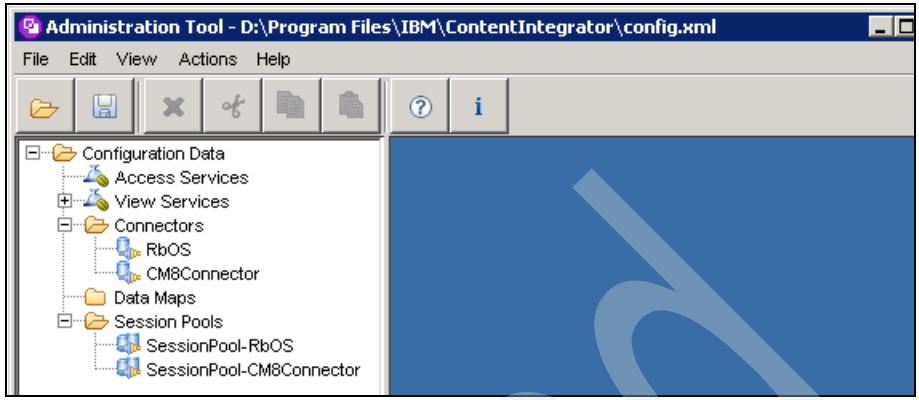



*Figure 5-15 Configured connector session pools*

## 5.4.5 Create the CFS federation administrator user

Create an Content Integrator user account to administer CFS:

1. Navigate to the *ICI_HOME*\bin directory.
2. Run the following command from a command line:

   ```
   run_sample commandline.SSOAdminTool
   ```

3. Select **1. Create Subject**.
4. Enter the *username* and the *password* when prompted.

Use this Content Integrator user account when you are prompted for the CFS Federation Administration application user during the installation of the CFS software.

## 5.4.6 Install CFS software

Install the CFS software on the cfsiciserver CFS/Content Integrator host, by following the instructions in the *IBM FileNet Content Federation Services Installation and Upgrade Guide*:

http://www.ibm.com/support/docview.wss?rs=3318&uid=swg27010328

Archived

At the conclusion of your installation of CFS, the CFS Federation Administration Tool (FedAdmin) will be deployed on an application server running on your cfsiciserver CFS/Content Integrator host. For this example, we assume FedAdmin is deployed in a WebSphere application server.

## Configure FedAdmin

After completing your installation of CFS, configure FedAdmin using the procedures that are described in the following sections.

### *Start FedAdmin*

Start the FedAdmin tool using the following procedure:

1. Start the CM8 RMI proxy connector on CM8 host (cm8server) by executing the batch file:

   `ICI_HOME\bin\RMIBridge.bat`

2. Start the P8 CM RMI proxy connector on the CFS/Content Integrator host (cfsiciserver) by executing the batch file:

   `ICI_HOME\bin\RMIBridge.bat`

3. Start the Content Integrator Configuration and SSO services on the CFS/Content Integrator host (cfsiciserver) by executing the batch file:

   `ICI_HOME\bin\VeniceBridgeServices.bat`

4. Log in to the FedAdmin application by entering the URL:

   `http://cfsiciserver:9080/FedAdmin`

### *General Configuration*

After logging into the FedAdmin tool, you are presented with the **General Configuration** tab, as shown in Figure 5-16 on page 226.

Archived

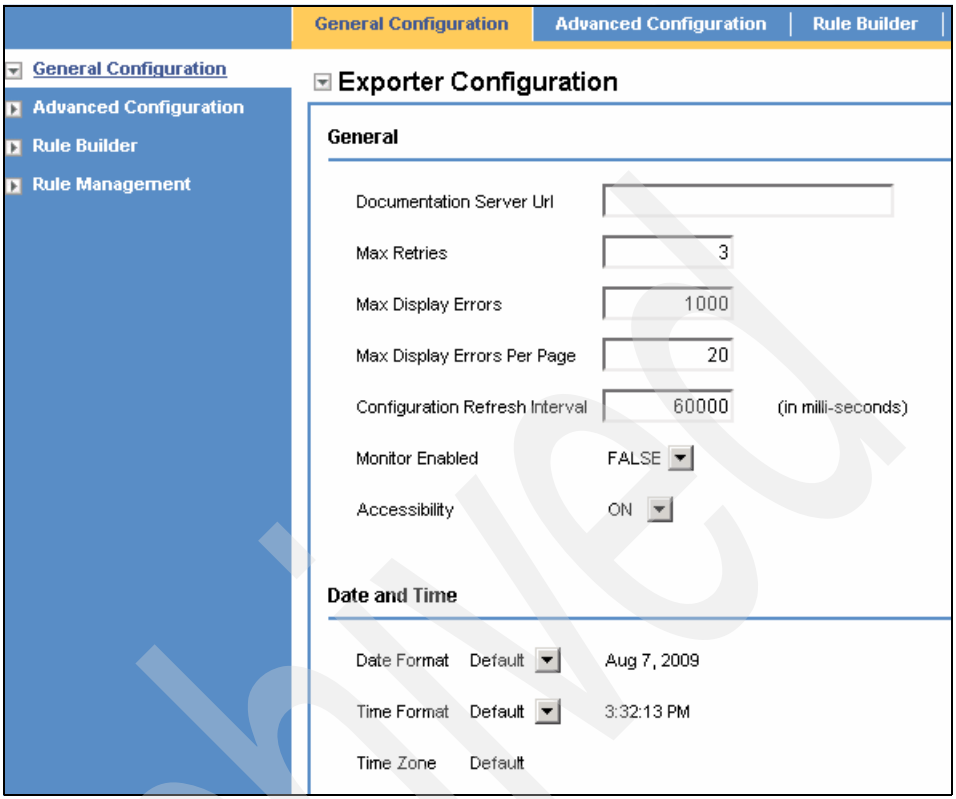



*Figure 5-16 FedAdmin general configuration*

The defaults on this tab are usually acceptable for most users. For a description of each of the fields on this tab, see the *IBM FileNet Content Federation Services Help:*

http://www.ibm.com/support/docview.wss?rs=3318&uid=swg27010328

### *Advanced Configuration*

Select the **Advanced Configuration** tab, as shown in Figure 5-17 on page 227.

Archived

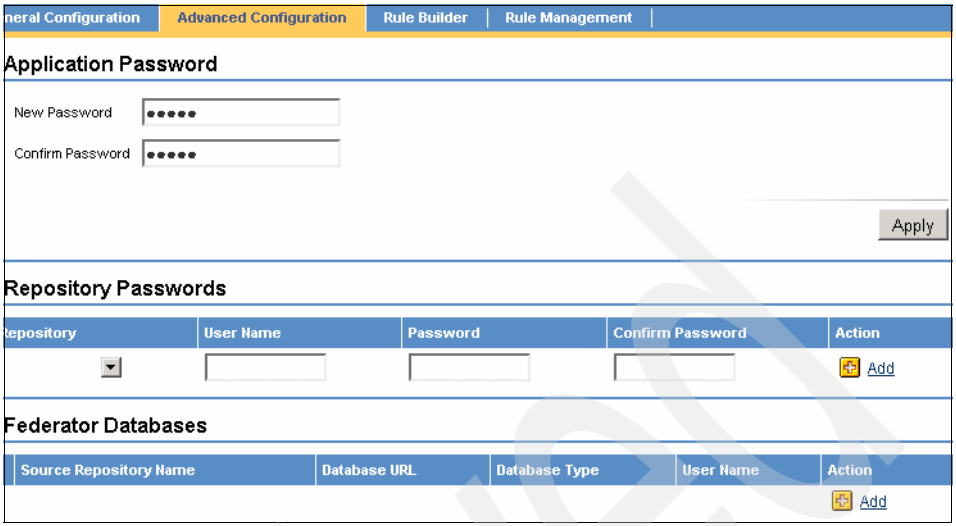



*Figure 5-17 FedAdmin advanced configuration*

This tab contains three sections:

- Application Password

  Use this section to change the password for logging into the FedAdmin tool.

- Repository Passwords

  Use this section to set the credentials for each of your Content Integrator connectors. These credentials are used by the FedAdmin tool and CFS exporter to automatically log in to the associated repository.

- Federator Databases

  Use this section to set up the connection parameters for your federation databases. You must have one federation database for each *source* repository, such as the CM8Connector federation database. The P8 CM repository, such as the RbOS P8 CM repository, is the *destination* repository, so it does not need a separate federation database.

  When you installed CFS, you created one federation database. This initial federation database is also referred to as the *master* federation database. If you only have one source repository, the master federation database is automatically used for it. So, you are *not* required to specify it. However, it is a good practice to enter the federation database information anyway.

We now step through the procedure for configuring the repositories and federation databases on this window.

Archived

### *Repository Passwords*

Configure the repository passwords using the following procedure:

1. From the Repository list, select your P8 CM connector, such as **RbOS**.
2. Enter the User Name and Password field values.
3. Click **Add**.
4. Now, test it by clicking **Test**, as shown in Figure 5-18.

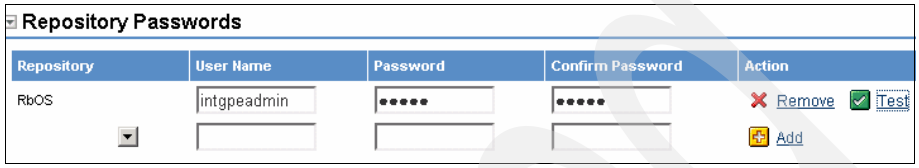



*Figure 5-18 Testing repository connections*

5. Repeat steps 1-4 for the CM8 connector, for example, CM8Connector.

### *Federator Databases*

Next, configure the federation databases using the following procedure:

1. Click **Add** to display the Federator Database Connection Configuration window, as shown in Figure 5-19.

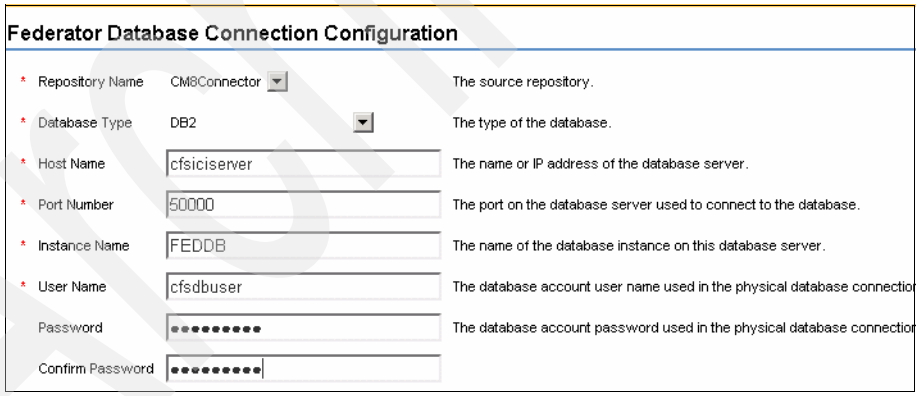



*Figure 5-19 Federation Database Connection Configuration window*

2. On Figure 5-19, select the CM8 source repository from the list. Note that we selected **CM8Connector**, because the federation database only needs to be configured for the *source* repository. We do *not* configure a federation database for the *destination* repository, such as the RbOS repository.
3. Fill in the remaining fields, as shown in Figure 5-19.

Archived

4. Click **Apply**.
5. Now, test it by clicking **Test**.

Log out of the FedAdmin tool for now. We will return later to create federation rules using FedAdmin after configuring P8 and setting up the data mapping between P8 and CM8.

## 5.4.7 Configure IBM FileNet Content Manager for CFS implementation (using Content Integrator)

The following list of tasks is a high-level view of the tasks that are necessary for configuring P8 for CFS implementation (using Content Integrator). See each task for detailed instructions. The tasks in this section are performed on the cm-cfsrb P8 CM host machine:

1. Configure Java Database Connectivity (JDBC) data sources for the fixed content device.
2. Create the Content Integrator fixed content device for the CM8 repository.
3. Create the Content Integrator fixed storage area for the CM8 repository.
4. Enable the CFS Import Agents.
5. Create a folder for Content Integrator federated documents from the CM8 repository.

In order to properly test the connections that you create in this section, make sure that you start the following Content Integrator processes if they are not already running:

1. Start the CM8 RMI proxy connector on the CM8 host (cm8server) by executing the batch file:

   `ICI_HOME\bin\RMIBridge.bat`

2. Start the P8 CM RMI proxy connector on the CFS/Content Integrator host (cfsiciserver) by executing the batch file:

   `ICI_HOME\bin\RMIBridge.bat`

3. Start the Content Integrator Configuration and SSO services on the CFS/Content Integrator host (cfsiciserver) by executing the batch file:

   `ICI_HOME\bin\VeniceBridgeServices.bat`

### Configure JDBC data sources for the fixed content device

Before the Content Integrator fixed content device can be created, two JDBC data sources must be created in the P8 CM application server. These two data sources must point to the federation database for the CM8 source repository. If

you were federating documents from multiple source repositories, you create two data sources per federation database source repository. See Figure 5-20.

| You can administer the following resources: | | | | | | |
|---|---|---|---|---|---|---|
| ☐ | ICICM8DS | ICICM8DS | Cell=cm-cfsrbNode01Cell | JDBC provider for DB2 | DB2 Universal Driver Datasource | |
| ☐ | ICICM8DSXA | ICICM8DSXA | Cell=cm-cfsrbNode01Cell | JDBC provider for DB2 (XA) | DB2 Universal Driver Datasource | |

*Figure 5-20 JDBC data sources for CM8 federation database*

See the *FileNet P8 Platform Installation and Upgrade Guide* for more information about how to create JDBC data sources:

http://www.ibm.com/support/docview.wss?rs=3278&uid=swg27010422

## Create the Content Integrator fixed content device

The Content Integrator fixed content device (FCD) is a P8 object that points to a source repository using Content Integrator. In our example, we point to a CM8 source repository. If you were federating documents from multiple Content Integrator source repositories, you create a separate FCD for each source repository. Follow these steps to create an Content Integrator FCD:

1. On the P8 CM host machine, such as cm-cfsrb, launch Enterprise Manager (EM) and log in as a user with administrator privileges.
2. Right-click **Fixed Content Devices** → **New Fixed Content Device**.
3. Assign a name to the FCD. Choose a name that indicates to which Content Integrator source repository you are connecting. We chose `ICI_CM8_FCD`. Click **Next**.
4. From the Type list on the top of the window, choose **Federated Device (Content Integrator)**.
5. For the remaining configuration parameters, set them up as shown in Table 5-2 on page 231.

Archived

*Table 5-2 Content Integrator FCD properties*

| Attribute name | Example value | Comments |
|---|---|---|
| FCP Pool Idle Timeout Seconds | 600 (default) | Enter the number of seconds that an idle connection can remain in the connection pool before being eligible to be closed. |
| FCP Pool Preferred Size | 5 (default) | Enter the minimum size to which the connection pool will be allowed to shrink after the idle timeout has expired. |
| FCP Pool Max Wait Seconds | 5 (default) | Enter the maximum number of seconds to wait for a connection after the limit specified by the “FCP Pool Max In Use” limit has been reached. |
| FCP Pool Max In Use | -1 (no limit) | Enter the maximum number of concurrent connections allowed for the FCD. If your source repository limits the number of concurrent sessions, set this parameter to the same value as your source repository limit. If you run multiple P8 servers in a farm, this value multiplied by your number of P8 servers must not exceed the limit imposed by the source repository. |
| Source Repository Name | CM8Connector | This name must be the same name as the value entered on the Content Integrator Administration Tool and FedAdmin for this source repository. |
| CFS User name | cfsuser | Enter the CM8 user that you created in “Create the CFS user” on page 206. |

Archived

| Attribute name | Example value | Comments |
|---|---|---|
| CFS password | | Enter the password that you created for cfsuser in CM8. |
| Federation JNDI Data Source | ICICM8DS | Enter the data source name that you created in "Configure JDBC data sources for the fixed content device" on page 229. |
| Federation JNDI XA Data Source | ICICM8DSXA | Enter the XA data source name that you created in "Configure JDBC data sources for the fixed content device" on page 229. |
| Federation Update Has Priority | `False` (default) | When true, changes to properties from the source repository are always applied, even if changes made by P8 applications on the federated document are more current. |
| Content Integrator URL | `rmi://cfsiciserver:1250/ConfigurationServer` | Enter the URL to the ConfigurationServer started by the *`ICI_HOME`*`\bin\VeniceBridgeServices.bat` batch file. If you run multiple configuration servers on separate hosts for High Availability, use a comma-separated list to list each of these servers. |

**Tip:** Do not use the Tab key to move to the next field, because it will cancel the wizard. Use the up arrow and down arrow keys instead.

See Figure 5-21 on page 233 for our example.

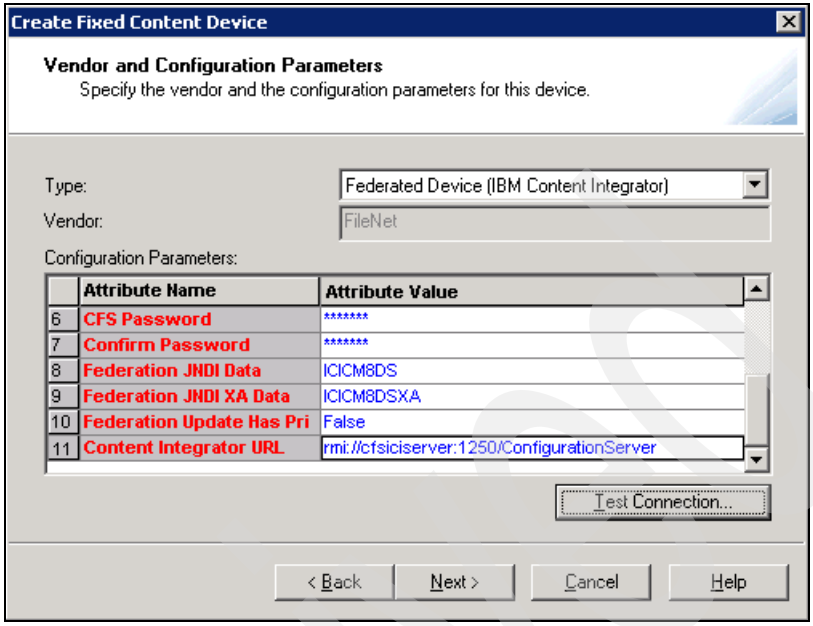



*Figure 5-21 Content Integrator fixed content device parameters*

6. Click **Test Connection** to verify that you can connect to the Content Integrator CM8 source repository.
7. Click through the rest of the wizard until it finishes and you receive a success message.

## Create the Content Integrator fixed storage area

In this section, we create a fixed storage area that uses the FCD that we created in the previous section.

> **Fixed storage areas:** Unlike fixed content devices, which are available to the entire P8 system, a fixed storage area is specific to a single object store.
>
> For more information about fixed storage areas, see **Help on FileNet Enterprise Manager Administration tool → Content Storage → Fixed Storage Areas**.

Follow these steps:

1. In IBM FileNet Enterprise Manager, right-click the name of the object store that you have identified as the target repository for the CM8 federated documents.

2. Click **New** → **Storage Area**.
3. Click **Next** at the welcome panel.
4. When prompted to select a site, click **Next** for Initial Site.
5. Assign a name to the storage area, and click **Next**. We chose the name `ICI_CM8_SA`.
6. For storage area type, select **Fixed Storage Area**, and click **Next**.
7. For Fixed Content Device, select the FCD that was created in the previous section, such as **ICI_CM8_FCD**.
8. For the Staging Area Path, specify an existing folder name on the P8 CM host machine, as shown in Figure 5-22.

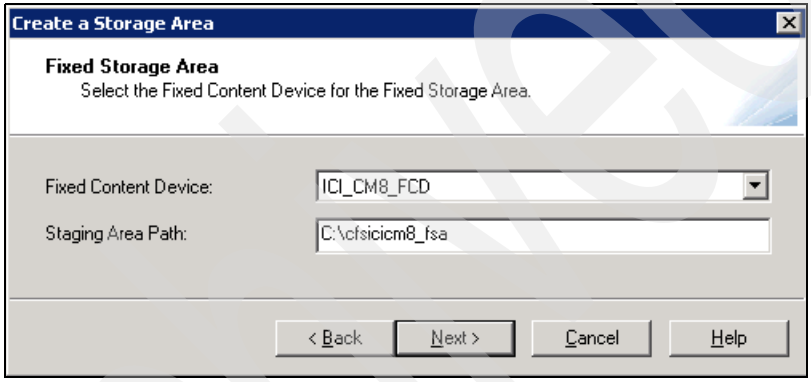



*Figure 5-22 Fixed storage area creation wizard*

> **If running in a clustered environment:** In this example, the staging area is created on a local drive. However, if the P8 CM server is running in a clustered environment, the staging area needs to be a shared directory on a local or remote server, which is accessible by all P8 CM servers using this same path.
>
> The requirement for the directory that is used for the staging area is the same as the requirement for the directory that is used for a file storage area. The directory must be capable of performing fast I/O operations with a significant amount of concurrency.

9. Click **Next** on the Select the Size Parameters of the Storage Area window.
10. Click **Next** on the Specify Storage Policy window.
11. Click **Finish** to complete the Create a Storage Area wizard.

## Enable CFS Import Agent

Enable the CFS Import Agent on P8 using the following steps:

1. In IBM FileNet Enterprise Manager, right-click the top level P8 domain, and click **Properties**.
2. Select the **CFS Import Agent** tab.
3. Select **Enable Dispatcher**. Your settings will look similar to Figure 5-23.

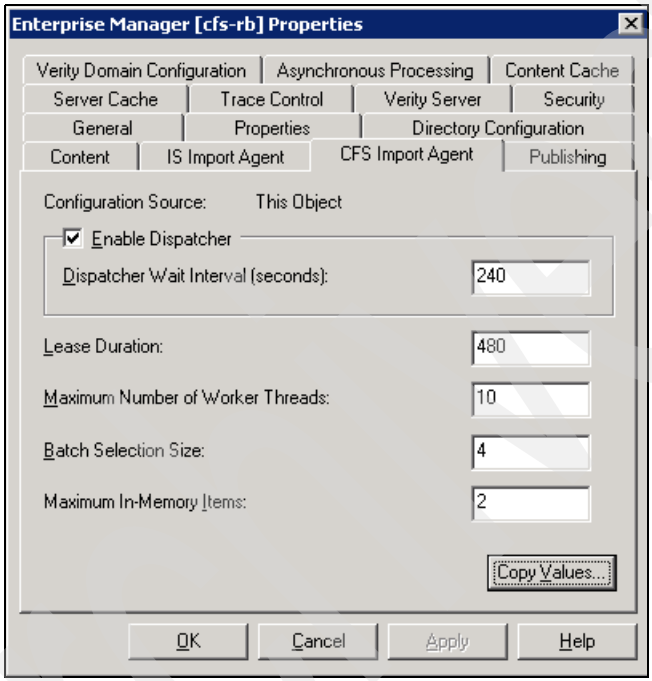



*Figure 5-23 CFS Import Agent tab*

4. Click **OK** to save your changes.

## Create a folder for CM8 documents

This procedure is optional. If you want CM8 documents to be filed in a specific folder in P8 during federation, use the following steps:

1. In IBM FileNet Enterprise Manager, select the object store that you have identified as the target repository for CM8 federated documents.
2. Next, navigate to **Root Folder**.
3. Right-click, and select **New Sub Folder**.

4. When prompted for a Folder Name, choose a name that indicates the Content Integrator source repository from which these documents originated. We type `CFS_ICI_CM8`.
5. Click **Create** to finish. Your folder looks similar to Figure 5-24.

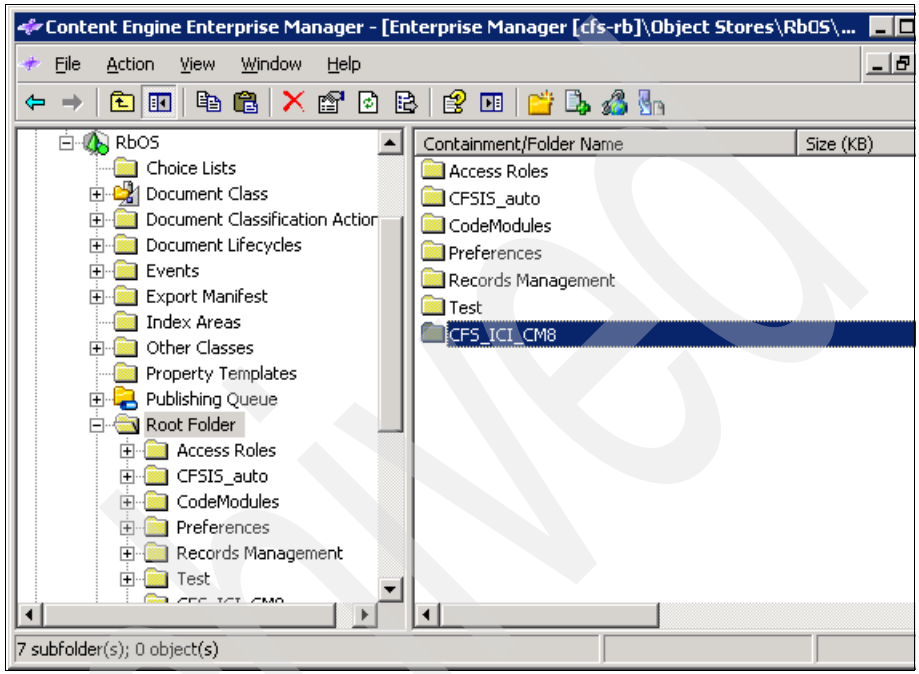



*Figure 5-24 Folder for CM8 federated documents*

## 5.5 Federate CM8 documents to FileNet P8

The following list is a high-level view of the tasks for federating CM8 documents to P8 using Content Integrator. See each task for detailed instructions:

- Create CM8 item type to federate.
- Create P8 document class to represent the CM8 item type.
- Create a data map to map a CM8 item type and attributes to a P8 document class and properties.
- Create a federation rule to federate desired item types from a CM8 source repository.
- Federate a document.
- Examine the federated document in P8.

## 5.5.1 Create a CM8 item type to federate

In planning your CFS implementation (using Content Integrator) for CM8, you identified existing CM8 item types that you want to federate to P8. In our example, we create a new CM8 item type to federate. We later create a rule to cause any new CM8 documents of this item type to automatically federate into P8. We perform the procedures in this section on the cm8server CM8 host machine.

Follow these steps to create a new item type in CM8:

1. Start the CM8 System Administration Client, and log in as an administrator user.
2. Navigate to **Data Modeling** → **Item Type** under your CM8 library server.
3. Right-click, and select **New** to create a new item type definition.
4. On the **Definition** tab, fill in the fields, as shown in Figure 5-25.

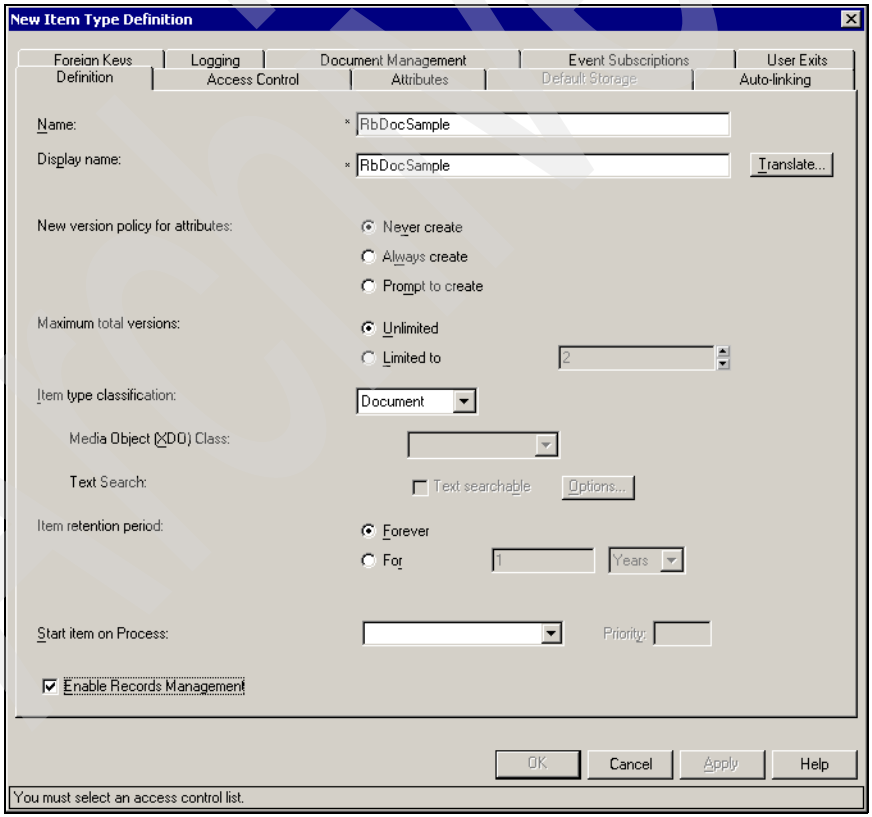



*Figure 5-25 Definition tab of RbDocSample item type*

As described in “CM8 required configuration options” on page 201, we set the following values on this tab (Figure 5-25 on page 237):

– New version policy for attributes, select **Never create**.

– Maximum total versions, select **Unlimited**.

– Item retention period, select **Forever**.

– Select **Enable Records Management**, because our example uses Records Management.

5. Click the **Access Control** tab. On this tab, we choose **PublicReadACL** for this item type, as shown in Figure 5-26. PublicReadACL is the ACL that we modified for the CFS user in “Modify access control lists for item types” on page 207.

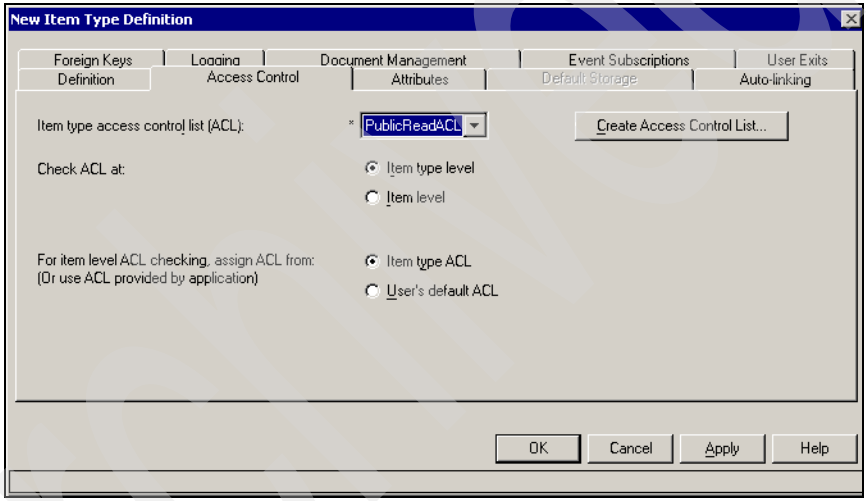



*Figure 5-26 Access Control tab of RbDocSample item type*

6. Click the **Attributes** tab. On this tab, we select several of the available attributes to include in our new item type, as shown in Figure 5-27 on page 239.

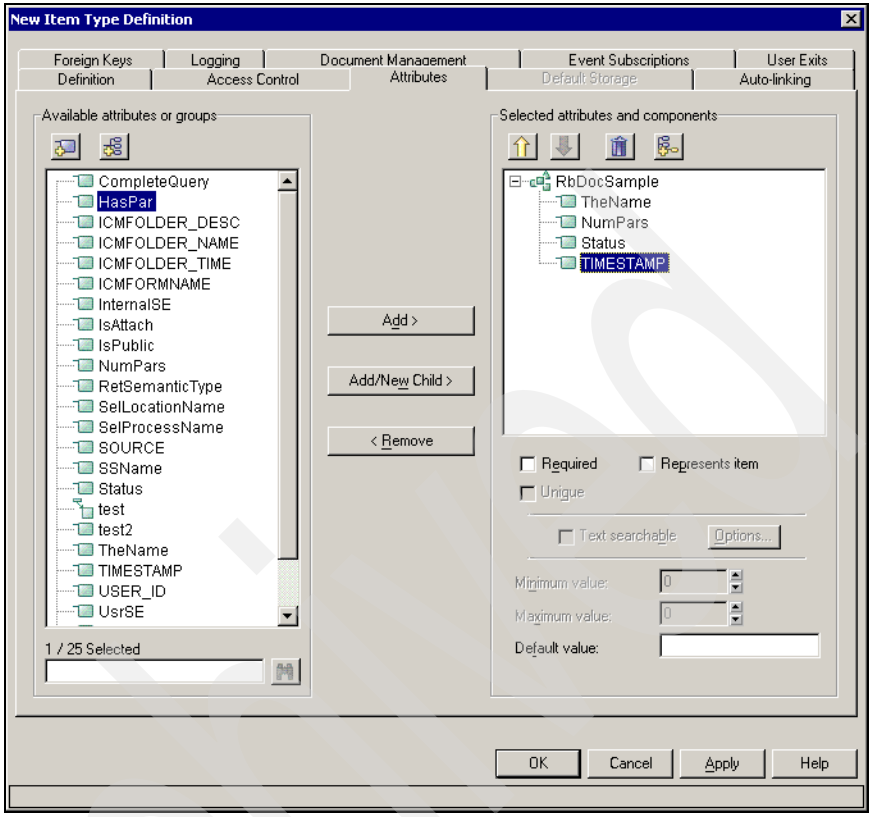



*Figure 5-27 Attributes tab of RbDocSample item type*

7. Click the **Document Management** tab to specify the document parts that are associated with this item type. We add one part type, as shown in Figure 5-28 on page 240.

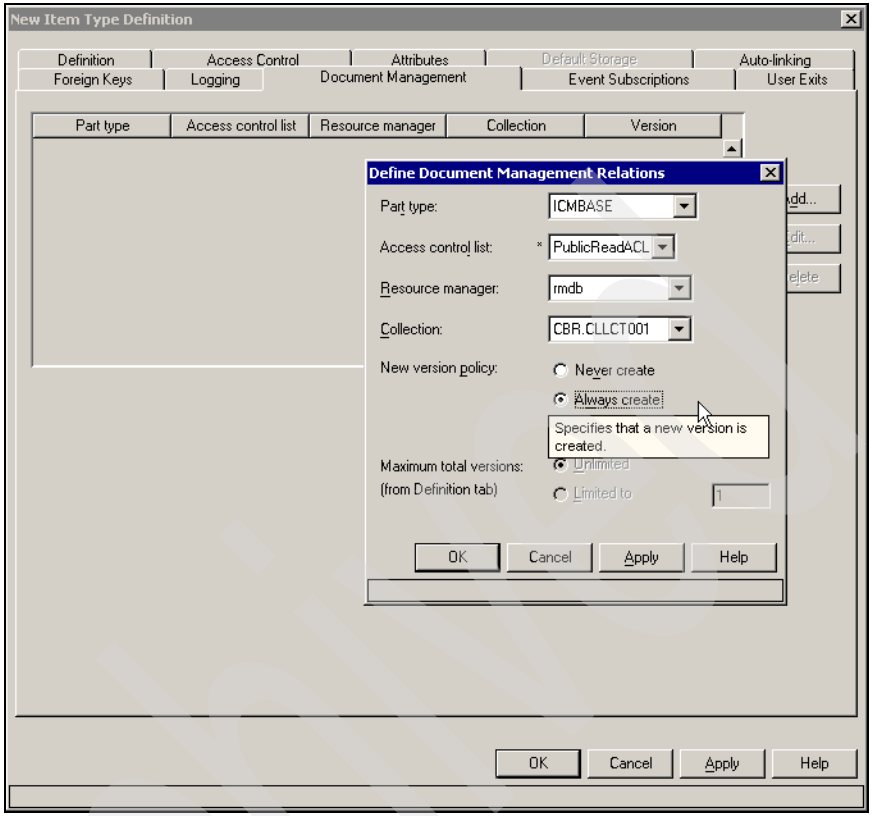



*Figure 5-28 Document Management tab of RbDocSample item type*

Note that we have chosen the Access control list PublicReadACL, which we have associated to our CFS user named cfsuser. Also, we have set the New version policy to **Always create**, as specified in “CM8 required configuration options” on page 201.

## 5.5.2 Create P8 document class to represent CM8 item type

Before we federate documents of the CM8 item type that we have created in the previous section, we create a corresponding P8 document class that we use to represent the CM8 item type. The procedures in this section are performed on the cm-cfsrb P8 host machine.

Use the following procedure to create a custom document class in P8:

1. Launch Enterprise Manager (EM), and log in as a user with administrator privileges.
2. Right-click the Document Class node in your object store, and select **New Class**.
3. Click **Next** on the welcome window.
4. Enter a name for the document class, and click **Next**. The name has to be unique within the object store. We entered `CFSICI_CM8_RbDocSample` for this example.
5. Click **Next** on the Select Properties window.
6. Click **Next** on the Select Property Attributes window.
7. Click **Next** on the Specify Content Storage Parameters window.
8. Click **Next** on the Configure Auditing window.
9. Click **Finish**.

### Add properties to the new custom document class

We now add properties to our custom document class to correspond to the CM8 item type attributes that we want to federate using the following procedure:

1. Right-click the CFSICI_CM8_RbDocSample custom document class that we created in the previous section, and select **Add Properties to Class**.
2. Click **Next** on the welcome window.
3. Click **New** to launch the Create a Property Template Wizard.
4. Click **Next** on the Create a Property Template welcome window.
5. Enter a name and select an appropriate data type. The property names must be unique within the object store. Table 5-3 shows a list of properties and their data types that we create for our example.

*Table 5-3 Add properties to class*

| P8 property name | Data type |
|---|---|
| CM8-Name | String |
| CM8-NumPars | Integer |
| CM8-Status | Integer |
| CM8-Timestamp | DateTime |

6. Click **Next** on the Select a Choice List window.

Archived

7. Select **Single**, and click **Next**.
8. Click **Finish**.
9. Repeat steps 2- 8 for each property that is shown in Table 5-3 on page 241.
10. After all of the properties are created, you return to the Add Properties to a Class Wizard. Click **Next** on the Select Properties window.
11. Click **Next** on the Select Property Attributes window.
12. Click **Finish** to complete the Add Properties to Class Wizard.

Figure 5-29 shows the properties of the CFSICI_CM8_RbDocSample P8 document class after following the previous steps. You can get this dialog by navigating to **Object Stores** → **RbOS** → **Document Class** → **CFSICI_CM8_RbDocSample**. Right-click, and select **Properties**. Then, click the **Property Definitions** tab.

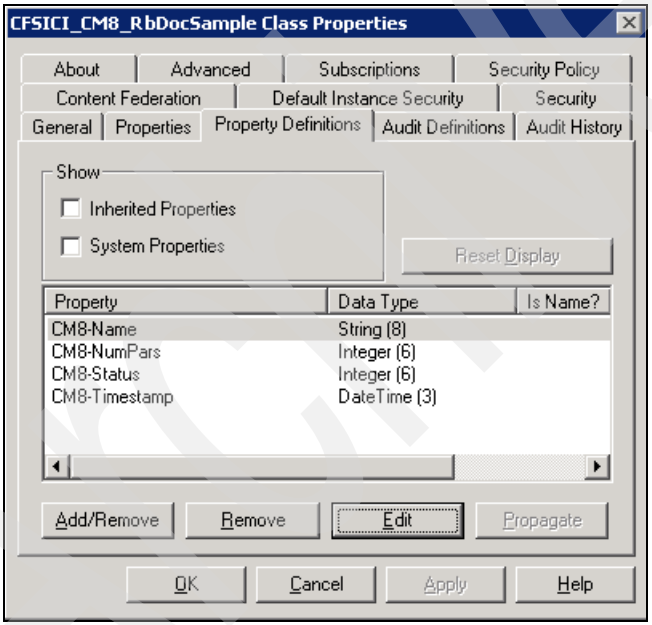



*Figure 5-29 Custom properties of example document class*

## Verify the P8 symbolic names of custom properties

After you finish creating the P8 properties for your new document class, verify the symbolic name that is assigned to each property. From the dialog in Figure 5-29, select a property and click **Edit**. We selected the CM8-Name property.

You will see a dialog similar to Figure 5-30. Notice that the display name is CM8-Name. However, the Symbolic Name value is CM8Name. You will need to

use the *symbolic* name value when you perform your data mapping in the next section.

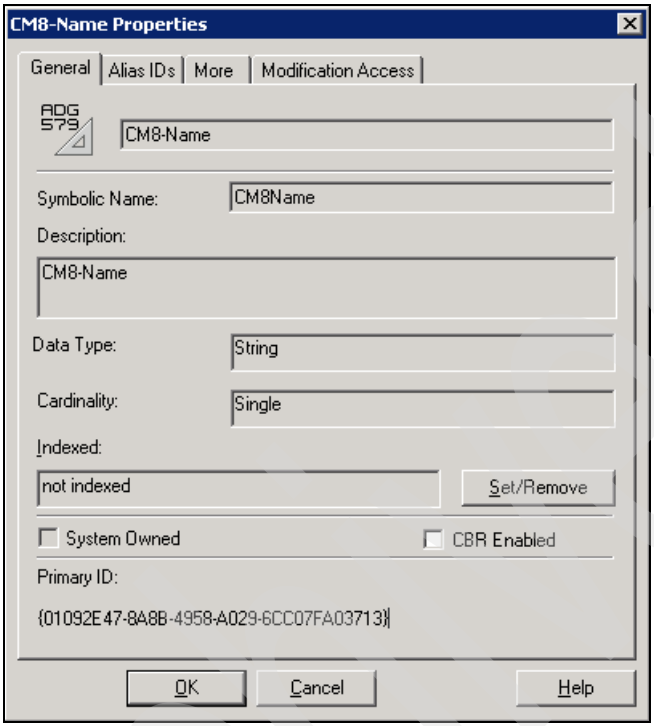



*Figure 5-30 Symbolic name of custom property*

## 5.5.3 Create a data mapping between CM8 and P8

Now that we have the CM8 item type and the P8 document class defined for this example, we can map the CM8 attributes to the P8 properties. We perform this mapping using the Content Integrator Administration Tool on the cfsiciserver CFS/Content Integrator host.

Prior to starting this data mapping activity, make sure that you start the following Content Integrator processes if they are not already running:

1. Start the CM8 RMI proxy connector on the CM8 host (cm8server) by executing the batch file:

   `ICI_HOME\bin\RMIBridge.bat`

2. Start the P8 CM RMI proxy connector on the CFS/Content Integrator host (cfsiciserver) by executing the batch file:

   `ICI_HOME\bin\RMIBridge.bat`

3. Start the Content Integrator Configuration and SSO services on the CFS/Content Integrator host (cfsiciserver) by executing the batch file:

   *ICI_HOME*\bin\VeniceBridgeServices.bat

Perform the data mapping using this procedure:

1. Start the Content Integrator Administration Tool.
2. Select **Configuration Data** → **Data Maps**.
3. Right-click, and select **New Data Map**.
4. Enter a name for this data map. We chose `CM8 to P8 Map`, as shown in Figure 5-31.

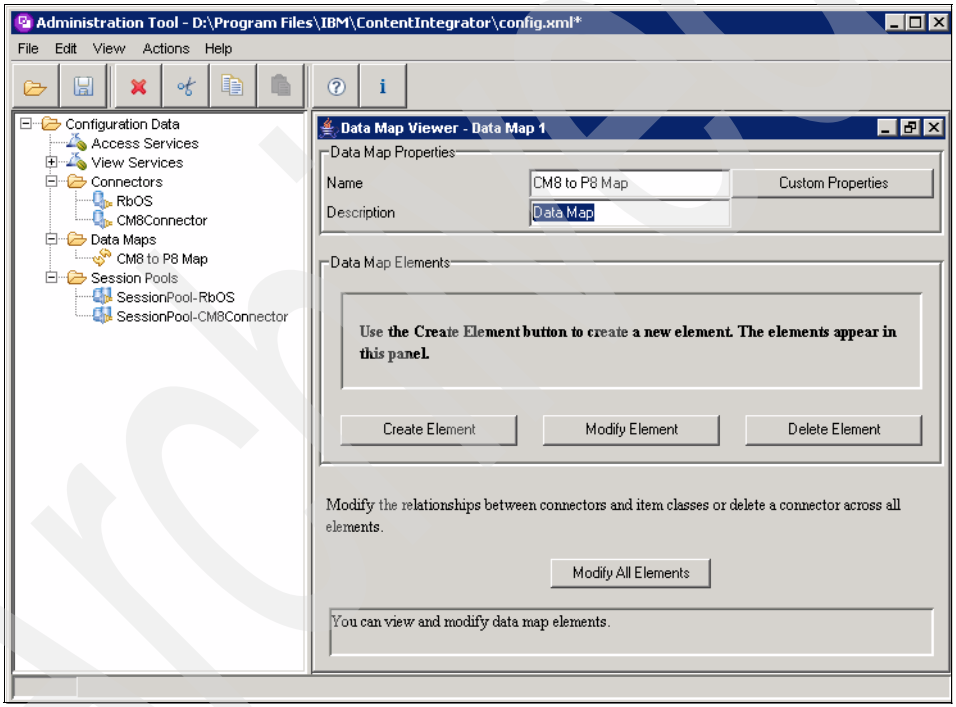



*Figure 5-31 Create CM8 to P8 Map*

5. Click **Create Element** to display the Data Map Element Editor.
6. Enter an Element Name value that corresponds to the *symbolic* name of the P8 property that you are mapping from the CM8 item type. Do *not* use the *display* name of the P8 property for this field. Refer to “Verify the P8 symbolic names of custom properties” on page 242 to validate the symbolic names of the P8 properties that you map.

We first map the CM8-Name property from our P8 document class. Note that this property has a display name of CM8-Name, but a symbolic name of CM8Name. So, we enter `CM8Name` for this field.

7. Next, we select the source repository connector, **CM8Connector**, in the Select a Connector section.
8. Then, we select our example CM8 item type, **RbDocSample**, in the Select an Item class section.
9. Finally, we select the property from this item type that we are mapping to the CM8Name P8 property. We chose the **RbDocSample/TheName** CM8 property.
10. The Data Map Element Editor looks similar to Figure 5-32.

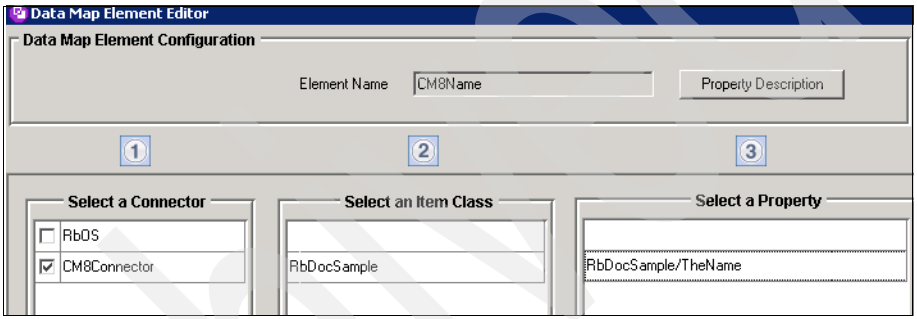



*Figure 5-32 Data map for CM8Name P8 property*

11. Click **OK** to complete the mapping for this P8 property.

> **Important:** Do *not* select the P8 connector on this form to specify the P8 Item Class and Property that map to this CM8 Item Class and Property. The P8 property is implicit in the Element Name field for this mapping. The P8 document class part of this mapping is determined by the federation rule that you will create in 5.5.4, “Create federation rules” on page 247.

12. Repeat steps 5 - 11 for each of the custom attributes of the CM8 item type, as shown in Table 5-4 on page 245.

*Table 5-4 Mapping of P8 custom properties*

| P8 property | CM8 property |
|---|---|
| CM8Name | RbDocSample/TheName |
| CM8NumPars | RbDocSample/NumPars |
| CM8Status | RbDocSample/Status |

| P8 property | CM8 property |
|---|---|
| CM8Timestamp | RbDocSample/TIMESTAMP |

13. Next, we map several CM8 system properties to P8 properties. CM8 system properties are not displayed on the CM8 System Administration Tool. However, they are available in the list when selecting a property in the Data Map Element Editor.

14. Map the P8 `DateLastModified` property to the CM8 system attribute **LASTCHANGEDTS**, as shown in Figure 5-33. As described in “Update” on page 195, adding a mapping for the P8 DateLastModified property allows CFS to detect which property update is newer if a property for a federated document is updated by both P8 and CM8 applications. If you do not create a mapping for the P8 DateLastModified property, the property updates that are made in the source repository will always overwrite any property updates that are made by a P8 application.

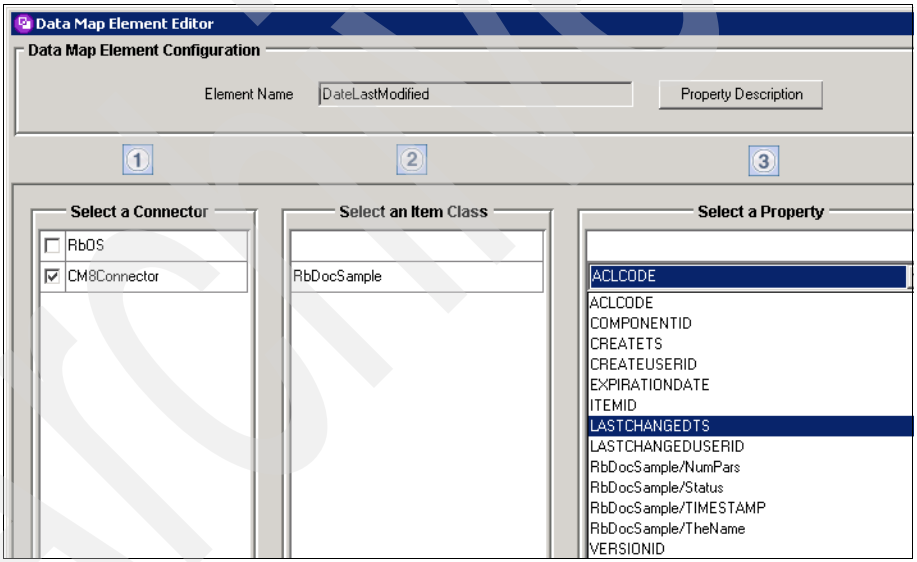



*Figure 5-33 Data map for DateLastModified P8 property*

15. In a similar manner, we map the P8 property **DocumentTitle** to the CM8 system property **ITEMID**, although, we can choose any CM8 property for this mapping. Because the P8 DocumentTitle property is used as the default name property in many P8 applications, such as WorkplaceXT, we recommend that you always create a mapping for it using a property in the source repository with a similar function.

16. After we map all of the properties, the Data Map Elements section of the Data Map Viewer looks similar to Figure 5-34.

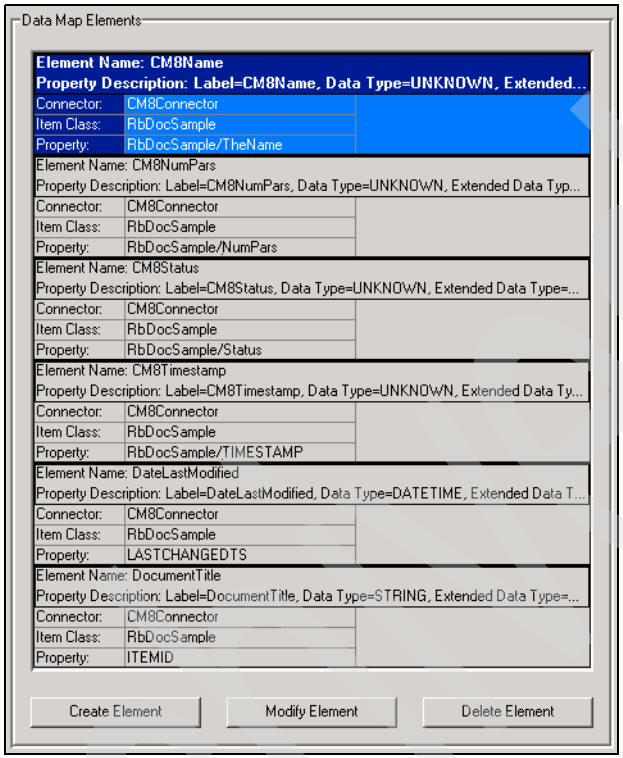



*Figure 5-34 Completed data map*

17. Save your configuration in the Content Integrator Administration Tool, and exit the application.

18. Restart the VeniceBridgeServices script to activate these Content Integrator configuration changes.

## 5.5.4 Create federation rules

*Federation rules* are used by the CFS exporter to select the documents to federate from the source repository and to apply the selected mapping. You create federation rules using FedAdmin. The procedures in this section are performed on the cfsiciserver CFS/Content Integrator host machine.

Prior to creating any federation rules, make sure that you start the following Content Integrator processes if they are not already running:

1. Start the CM8 RMI proxy connector on CM8 host (cm8server) by executing the batch file:

   *ICI_HOME*\bin\RMIBridge.bat

2. Start the P8 CM RMI proxy connector on the CFS/Content Integrator host (cfsiciserver) by executing the batch file:

   *ICI_HOME*\bin\RMIBridge.bat

3. Start the Content Integrator Configuration and SSO services on the CFS/Content Integrator host (cfsiciserver) by executing the batch file:

   *ICI_HOME*\bin\VeniceBridgeServices.bat

**Important:** You *must* restart VeniceBridgeServices whenever you make a change to the data mapping in the Content Integrator Administration Tool.

## Create the federation rule

Create federation rules with the FedAdmin tool using the following procedure:

1. Log in to the FedAdmin application by entering the URL in a browser:

   http://cfsiciserver:9080/FedAdmin

2. Click the **Rule Builder** tab.
3. Enter a Rule Name. We chose RbDocSampleRule.
4. Click the Data Map list to select the data map that you created in the previous section. We chose **CM8 to P8 Map**.
5. Choosing a Data Map causes the Repository Name list to be populated with the repositories that are associated with that data map. So, select the *source* repository for this data map. We chose **CM8Connector**.
6. Click the Document Class list, and select the CM8 item type to federate. Only CM8 documents of this item type will be federated using this rule we are building. We chose **RbDocSample**.
7. We have the option now of selecting additional criteria that documents of the chosen CM8 item type must match to be federated. We chose to federate all of the documents of the RbDocSample item type that we selected in the previous step. So we left the Full-Text Criteria and the Property Criteria fields blank.
8. Set the Max Results field if you want your rule to limit the number of documents to federate each time that the rule executes. This limit is typically only useful when testing your rule using the Search button as described in the next step.

If you are only using Max Results to test your rule, remember to remove its value before saving your rule and scheduling it to be run by the CFS exporter. If you leave a value for this field, every time that the exporter executes this rule, it will only federate up to this maximum. If there are more documents that match the selection criteria, those documents will not be federated until the next time the exporter runs this rule. So, if the exporter runs this rule only one time a day, it can take an extremely long time for all of your documents to be federated.

9. Now, click **Search** to see how many documents in the CM8 source repository match our current search criteria. As seen in Figure 5-35, our search results in one matching item.

   If you have *not* set Max Results, the search results are limited to the first *chunk* of items that match the selection criteria. This approach helps prevent long running queries from this window if an extremely large number of items match the selection criteria. You set the *chunk* size on the General Configuration tab under Federation Query settings.

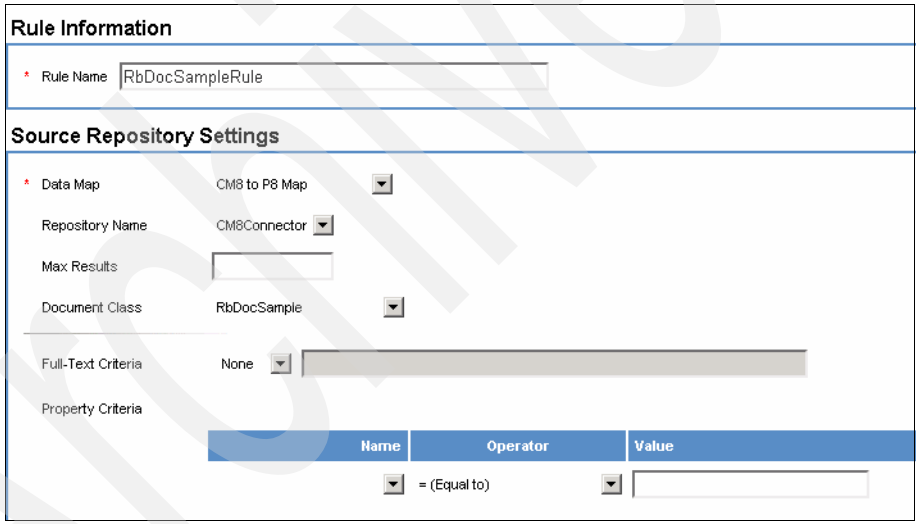



*Figure 5-35 Rule Builder: Source Repository Settings*

10. Next, we set the Target Repository Settings by first selecting the object store. Click the **Browse** link next to the Object Store field, as shown in Figure 5-36.

Archived

*Figure 5-36 Rule Builder: Target Repository Settings*

11. The window that is shown in Figure 5-37 is displayed. If we select **RbOS** and click **Select**, our federated documents are unfiled in the target P8 object store.

*Figure 5-37 Target repository object store*

12. We choose to file these federated documents in the CFS_ICI_CM8 folder that we created in “Create a folder for CM8 documents” on page 235. So, we click the **RbOS** link. Then, we select **CFS_ICI_CM8**, as shown in Figure 5-38 on page 250. Click **Select** to save your choice.

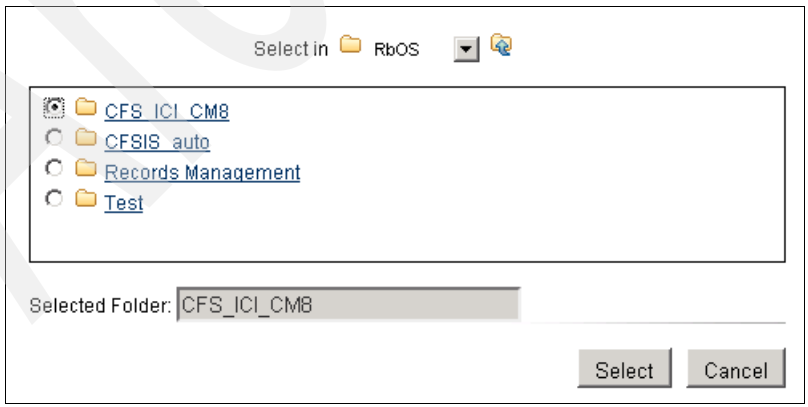



*Figure 5-38 Target repository folder*

Archived

**Default folder:** If you do not choose a target repository folder, documents are filed in the root folder of the object store.

13. Next, we click the **Document Class** list to select the P8 document class to which the CM8 item type needs to be mapped. We chose the **CFSICI_CM8_RbDocSample** document class. When finished, your Target Repository Settings section for the Rule Builder tab looks similar to Figure 5-39.

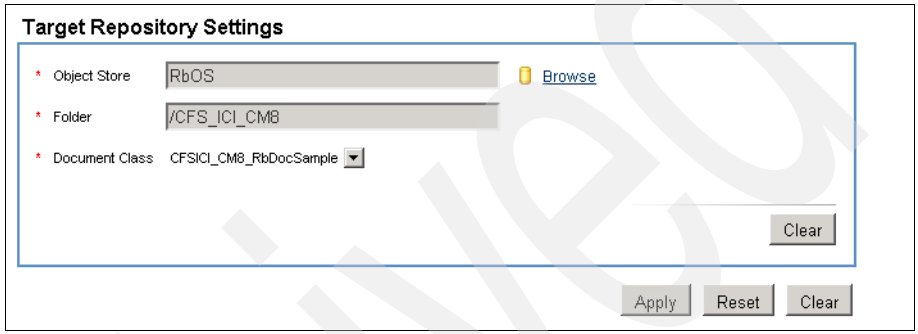



*Figure 5-39 Rule Builder: Target Repository Settings*

14. Click **Apply** to save your rule.

## View the newly created federation rule

Click the **Rule Management** tab to view the rule that you have created. When you create a new rule, it is automatically placed in Scheduled status, as shown in Figure 5-40.

The CFS exporter can start executing rules after they are in the Scheduled status.

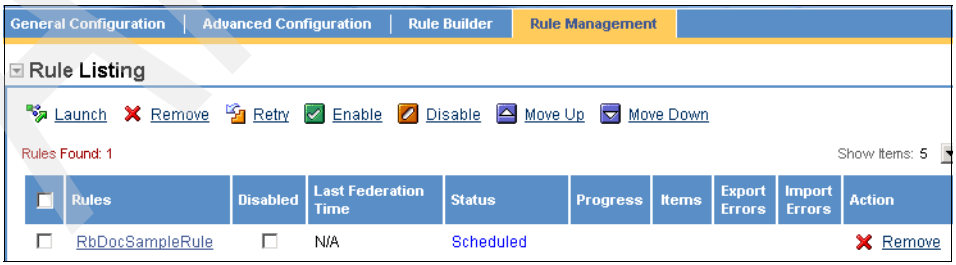



*Figure 5-40 Rule Management: Scheduled rule*

Archived

## 5.5.5 Federate a document

Now that we have our data mapping and federation rules defined, we can run a test to federate a CM8 document.

### *Create a new CM8 document*

To create a new CM8 document, perform the following steps:

1. Log in to the cm8server CM8 server.
2. Start the CM8 client application by clicking **Programs** → **IBM DB2 Content Manager Enterprise Edition** → **Client for Windows**.
3. Log in as a user that can create new documents in CM8.

> **CM8 privileges:** The CFS user that we created in "Create the CFS user" on page 206 was not given the CM8 privilege to create items. So, use an alternate CM8 user with this privilege to create new CM8 documents.

4. Select **File** → **Import** on the Client for Windows application.
5. Click **Add Files to Import>>** to select a file to import into the CM8 repository.
6. Click the Item Type list to select an item type that is configured for CFS, as we have done in 5.5.1, "Create a CM8 item type to federate" on page 237. We chose **RbDocSample**.
7. Fill in the attributes that are defined for this item type, as shown in Figure 5-41.

Archived

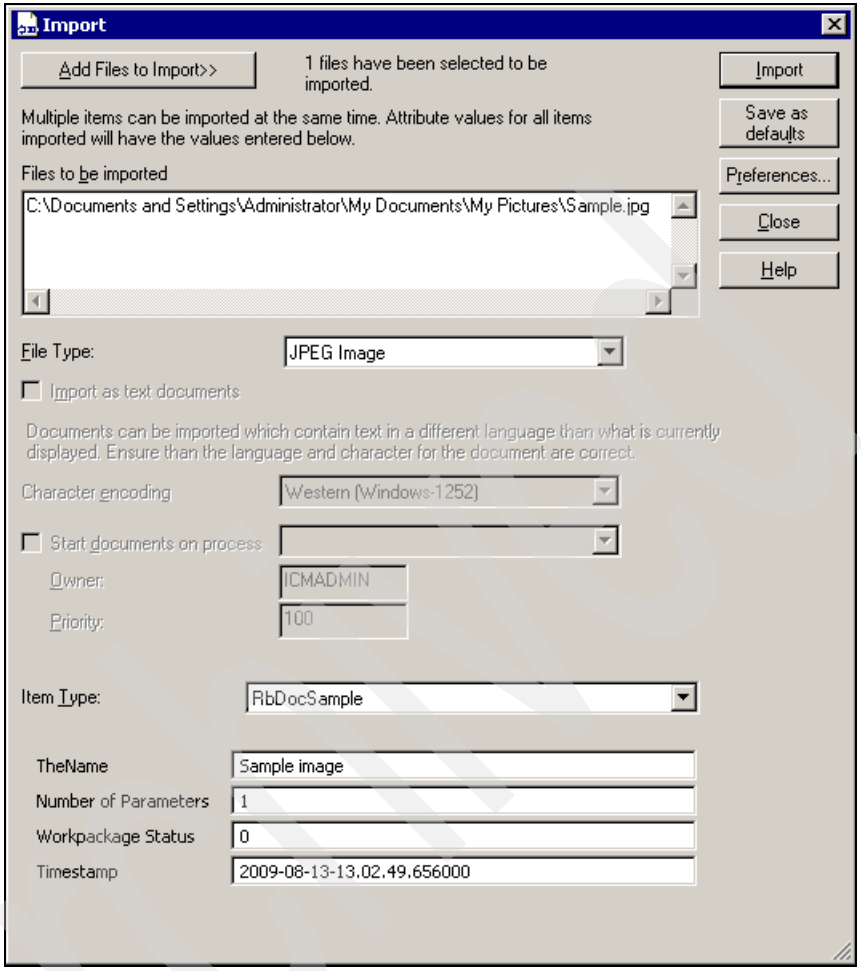



*Figure 5-41 Create new CM8 document*

8. Click **Close** to close the Import window.
9. Choose **File** → **Exit** to exit the CM8 client application.

## Monitor the CFS progress

Prior to exporting or monitoring the CFS progress, make sure that you start the following Content Integrator processes if they are not already running:

1. Start the CM8 RMI proxy connector on the CM8 host (cm8server) by executing the batch file:

   `ICI_HOME\bin\RMIBridge.bat`

2. Start the P8 CM RMI proxy connector on the CFS/Content Integrator host (cfsiciserver) by executing the batch file:

   *ICI_HOME*\bin\RMIBridge.bat

3. Start the Content Integrator Configuration and SSO services on the CFS/Content Integrator host (cfsiciserver) by executing the batch file:

   *ICI_HOME*\bin\VeniceBridgeServices.bat

Now, start the CFS exporter, if it is not already running on the CFS/Content Integrator host (cfsiciserver), by executing the batch file:

*CFS_HOME*\FedExporter\bin\run_exporter.bat

To monitor the CFS progress, perform the following steps:

1. Log in to the FedAdmin application by entering the URL in a browser:

   http://cfsiciserver:9080/FedAdmin

2. Click the **Rule Management** tab.
3. Examine the value of the Status column for the rule that we created in “Create the federation rule” on page 248. In Figure 5-42, the status for our example is Exporting. A normal federation run goes through five Status values: Scheduled, Processing, Exporting, Importing, and Federated.

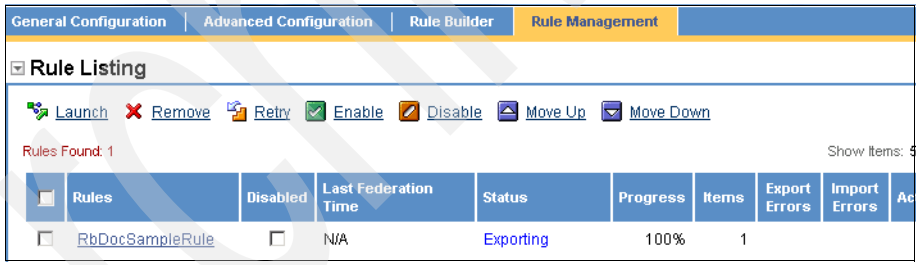



*Figure 5-42 CFS rule in the Exporting status*

4. Click **Refresh** periodically to watch the status of the federation rule change. When the status changes to Federated, as shown in Figure 5-43 on page 255, federation is complete. The value under the Items column shows the number of documents that were federated. In our case, we federated one document.

Archived

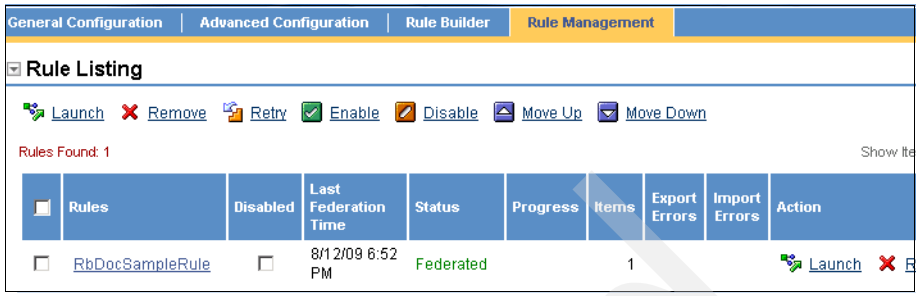



*Figure 5-43 CFS rule in Federated status*

d. Log out of FedAdmin.

## 5.5.6 Verify federated document

In the previous section, we federated one document. Now, we invoke a P8 application to view that document and verify that the properties of the P8 federated document are the same as the attributes of the CM8 source document.

Use the following procedure to verify the P8 federated document:

1. Log in to a P8 CM host, such as cm-cfsrb.
2. Launch IBM FileNet Enterprise Manager, and navigate to the folder containing your CM8 federated documents. We navigate to **Object Stores** → **RbOS** → **Root Folder** → **CFS_ICI_CM8**.
3. Locate your document by browsing the documents in this folder. Use the Created column as a hint to locating your document. Your federated document's creation time is close to the Federation Time shown on FedAdmin when your rule was run.
4. An alternative method of finding your federated document is to search for it using one of the mapped properties.
5. In IBM FileNet Enterprise Manager, navigate to the object store search folder. We navigate to **Object Stores** → **RbOS** → **Search Results**.
6. Right-click, and select **New Search**.
7. On the Select From Table field, choose the custom P8 document class for your federated document. We chose **CFSICI_CM8_RbDocSample.**
8. In the Criteria section of the form, use the custom properties of the document class to specify the known values of the source document. Our example in Figure 5-44 on page 256 specifies **CM8-Name** Equal To `Sample image`, which corresponds to the TheName attribute of the CM8 document that we created in Figure 5-41 on page 253.

Archived

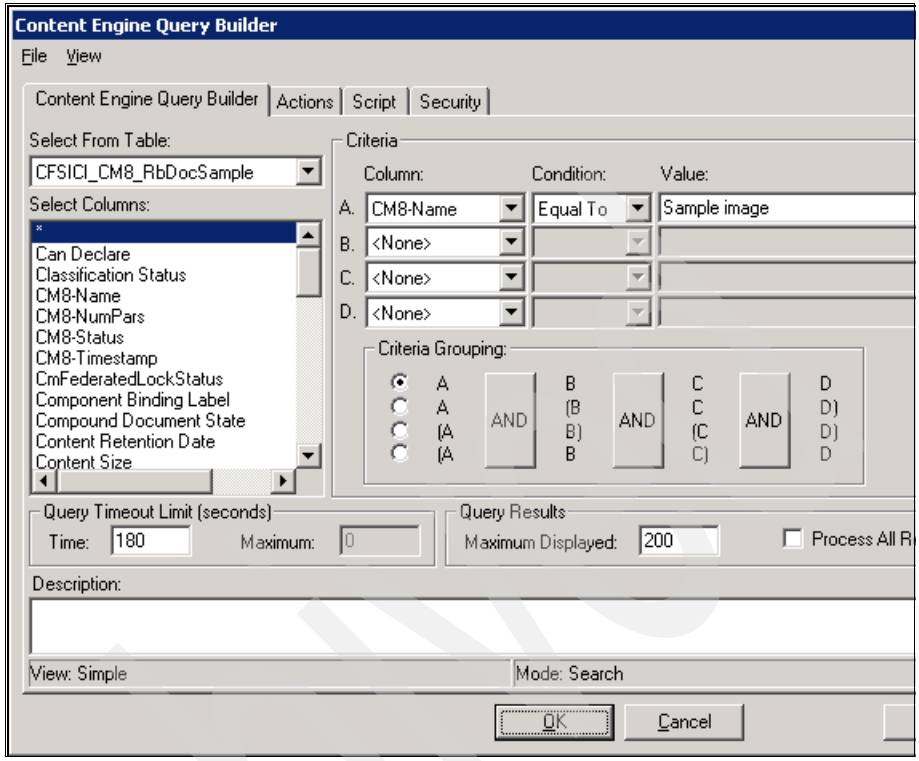



*Figure 5-44 Search criteria for federated document*

9. Click **OK** to execute the search.
10. Select the found document in the Search Results pane, right-click it, and select **Properties**.
11. Select the **Properties** tab, as shown in Figure 5-45 on page 257. The custom property values of this federated document need to be the same as the corresponding attributes of the CM8 source document that is shown on Figure 5-41 on page 253.

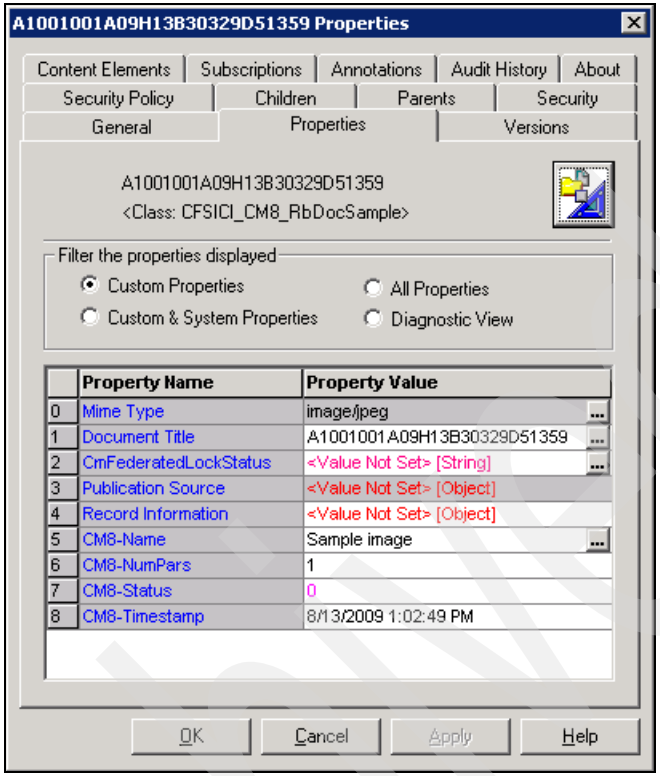



*Figure 5-45 Custom properties of federated document*

12. Select the **Content Elements** tab to view the content. The content of the P8 document needs to be the same as the content of the CM8 source repository document.

## 5.5.7 Schedule the federation rule to run automatically

Now that you have verified that your federation rule works, it is time to schedule it to run automatically.

### Setting up the default schedule

Log back in to FedAdmin and go to the **Rule Management** tab. We set up the default schedule using the Service Scheduler section, as shown in Figure 5-46 on page 258. Unless configured otherwise, all federation rules are scheduled to execute based on this default schedule.

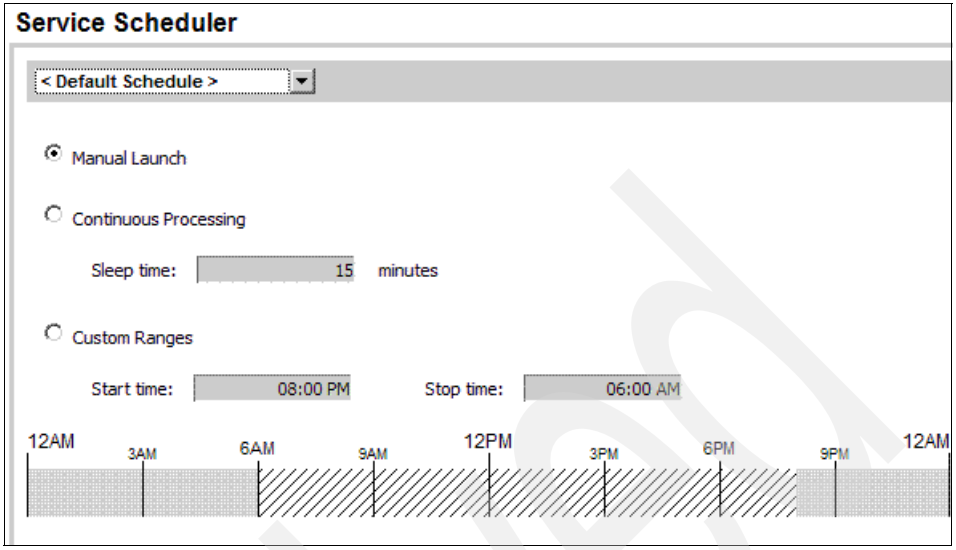



*Figure 5-46 Service Scheduler*

There are three scheduling options for a federation rule that you can choose:

- Manual Launch

  The rule is executed manually using the **Launch** action on the Rule Listing section.

- Continuous Processing:
  - Rules are automatically processed after *n* minutes of Sleep time from the end of the previous execution of the rule.
  - If you click Launch on the Rule Listing section, the rule is immediately processed (even during the sleep period).

- Custom Ranges:
  - The processing of the rules automatically starts at the Start time of each range and stops at Stop time of each range:
    - If the previous execution of a rule is complete when the next Start time is reached, a new federation is launched for the rule.
    - If the previous execution of a rule is *not* complete when the next Start time is reached, the previous federation is resumed.
  - Only one execution of a rule per range can be launched. Note that the execution of a rule can span two or more ranges.

Archived

– If the Launch action is selected on the Rule Listing section, the rule is immediately processed:
  - Even if it has already been processed in the current range
  - Even if we are outside of a range
– You can add multiple custom ranges by clicking the Add link and providing additional start and end times. The example in Figure 5-47 schedules federation rules to run twice daily, between 10:00 pm and 7:00 am and between 2:00 pm and 6:00 pm. Note that the first range spans a calendar day.

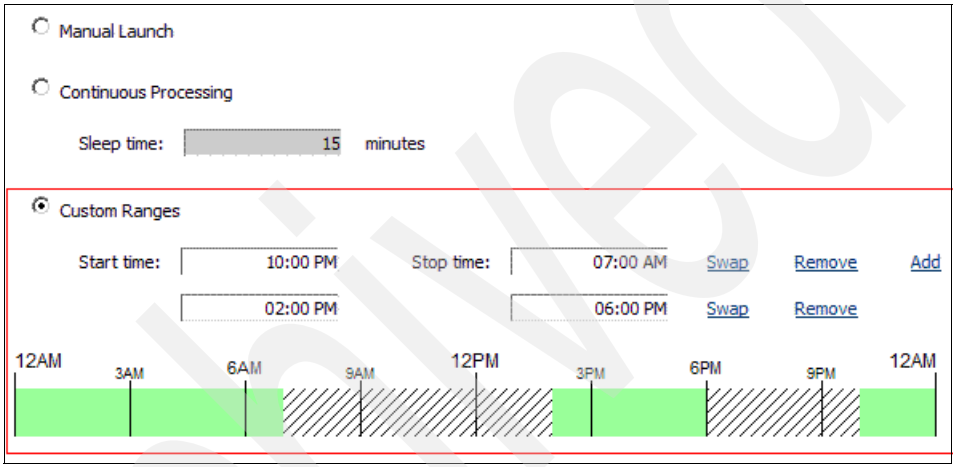



*Figure 5-47 Federation rule schedule with custom range*

## Setting up an individual rule schedule

If you do not want to schedule a rule using the default schedule, create an individual rule schedule. First, choose the federation rule from the list for which you want to customize a schedule, as shown in Figure 5-48 on page 260.

Archived

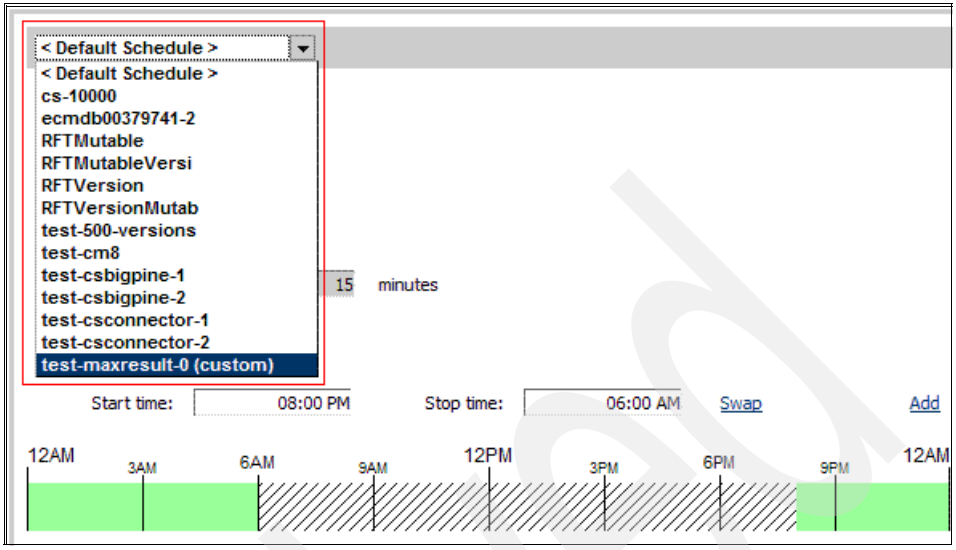



*Figure 5-48 Create a schedule for an individual rule*

Next, modify the schedule by selecting a separate scheduling option. The schedule options for an individual rule are the same as the options that are described in the previous section. See Figure 5-49.

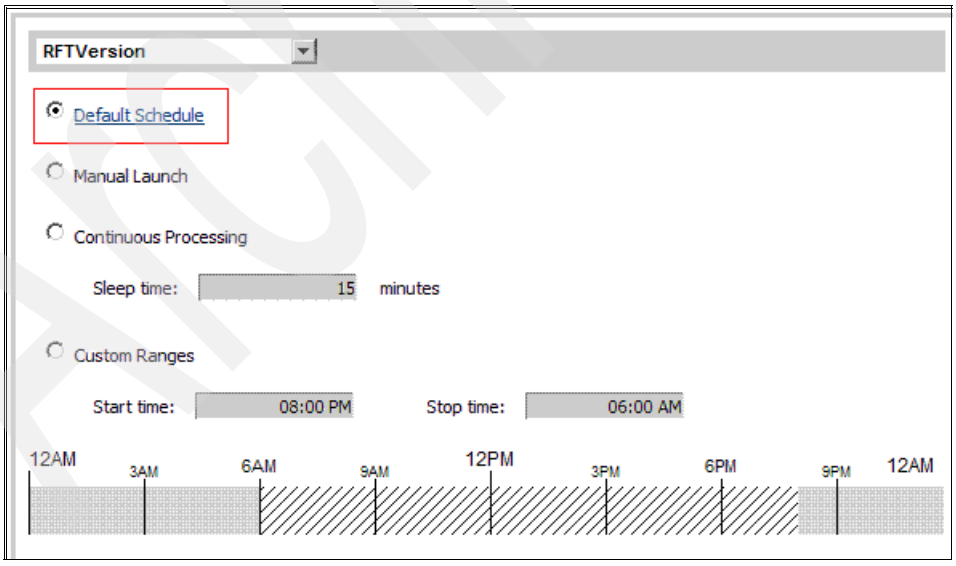



*Figure 5-49 Select scheduling option for an individual rule*

Archived

## 5.5.8 View federated documents in WorkplaceXT

During the CFS implementation (using Content Integrator), you can specify the P8 Content Manager folder where you want to store federated documents. If you do not specify a folder, the federated documents are stored in the root folder, which can be accessed through the FileNet Enterprise Manager application. You can locate the documents in the root folder by searching for them; however, you cannot browse for them using the WorkplaceXT interface.

During the configuration of CFS implementation (using Content Integrator), you can make federated documents virtually indistinguishable from documents that are added to P8 natively. The document class and the associated metadata can be identical with the only visible indicator of a federated document being the lack of a value in the Size property, as shown in Figure 5-50. Figure 5-50 shows three documents, two of which were federated. You will notice that the two federated documents lack a value in the Size property.

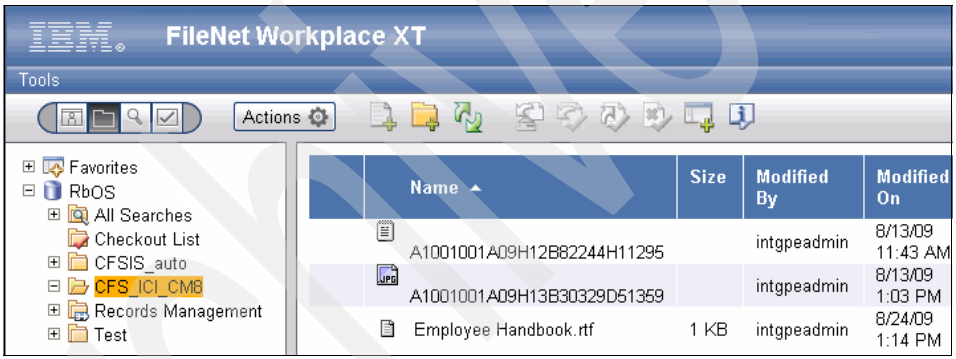



*Figure 5-50 Viewing federated documents in P8 WorkplaceXT*

In certain situations, it is beneficial to provide users with an easy way to find federated documents. There are multiple methods that can provide this ability. These methods are the most common:

- Create a exclusive document class for federated content. Users can create document class-specific queries to locate federated content.
- Create an index property that gets populated automatically during the federation process. Users can perform searches based on this property value. *Note that you must have created a mapping from a source repository property to this P8 index property to get this value populated during federation.*

Archived

### Federating and viewing CM8 MO:DCA files

To view MO:DCA documents, you must use WorkplaceXT. No support exists for viewing MO:DCA documents in Workplace or IBM FileNet Enterprise Manager. In WorkplaceXT, the download capability is disabled for MO:DCA documents, and the only access to MO:DCA content is through the WorkplaceXT viewing capability. In Workplace, the download capability is not disabled, but the download will not contain the correct background information for the MO:DCA document.

CM8 MO:DCA files are a composite of two entities:

- The `.mda` file representing the content
- The `.frm` file defining how the content is displayed

The combined presentation is the main content overlaying the forms. The `.frm` file is stored as an ICMFORMS item type in the CM8 repository. The ICMFORMS item class contains one single attribute called ICMFORMNAME.

To federate MO:DCA files, both the `.mda` files and `.frm` files must be federated. The federation of the `.mda` file is identical to the federation of any normal document. To federate the `.frm` file, you must define a separate rule to map the ICMFORMNAME to the Document Title property in P8. Typically, there are a small number of forms in the CM8 source repository, and you can use a single rule to federate all the forms into P8. Do not change the value of Document Title for the federated form files. If the Document Title property is changed, P8 cannot locate the form files, and you will not be able to view these MO:DCA files in WorkplaceXT correctly.

## 5.6 Troubleshooting

In this section, we provide troubleshooting information and tips. We cover common issues that you might encounter during CFS implementation (using Content Integrator).

### 5.6.1 Logs

A number of logs exist that can be consulted during troubleshooting. The logs are separated into FedAdmin logs and Exporter logs. CFS provides an IBM Support Assistant (ISA) add-on that can automate the data collection of these logs.

Archived

## FedAdmin logs

The FedAdmin logs are combined with the application server logs and can be found in the application server log directory. Depending on the application server to which FedAdmin is deployed, the log files are located in one of the following folders:

- WebSphere

  `<WAS_HOME>\profiles\<FEDADMIN_PROFILE>\logs\<SERVER_NAME>`

- WebLogic 8.*x*

  `<BEA_HOME>\user_projects\domains\<FEDADMIN_DOMAIN>\<SERVER_NAME>`

- WebLogic 9.*x*

  `<BEA_HOME>\user_projects\domains\<FEDADMIN_DOMAIN>\servers\<SERVER_NAME>\logs`

- JBoss

  `<JBOSS_HOME>\server\<SERVER_NAME>\log`

## Exporter logs

The Exporter logs provide important information about the execution of each federation. They also help you to understand how the concurrency occurs during the federation of multiple rules matching a large number of items.

The Exporter log files are located in the `CFS_HOME\FedExporter\logs` directory.

The Exporter uses the log4j framework for logging. You can customize the log4j properties file to increase file size, increase the backup index, change the logging patterns, and more. The log4j properties file location is `CFS_HOME\FedExporter\conf\log4j.properties`.

You can change the Exporter logging level, as well as the log file name, using FedAdmin. To log the maximum amount of information, change the logging level to DEBUG. But, be aware that a DEBUG logging level can drastically reduce performance and can result in the generation of extremely large log files. Figure 5-51 shows the Exporter Configuration window.

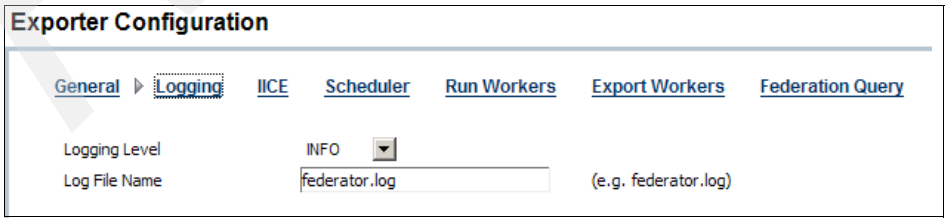



*Figure 5-51 Exporter Configuration window*

## Exporter log descriptions

Each log entry is prefixed by the name of the thread that produces the log.

[Scheduler]:

- This entry contains information about the status of each rule.
- An entry for each rule is generated periodically.
- The refresh interval can be changed from the Admin UI (select **General Configuration** → **General** → **Refresh Interval**).

[RunDispatcher]:

- This entry contains information about how the rules are dispatched to the run workers.
- This entry contains the state of the queue containing reserved rules that are ready to be dispatched and processed by the run workers.

[RunWorker-<*thread_number*>]:

- This entry contains information about the processing of the rules.
- This entry contains information about when the processing starts.
- This entry contains information about which query is executed.
- This entry contains information about how many items are retrieved.

[ExportDispatcher]:

- This entry contains information about how the batches are dispatched to the export workers.
- This entry contains the state of the queue containing reserved batches that are ready to be dispatched and processed by the export workers.

[ExportWorker-<*thread_number*>]:

- This entry contains information about the processing of the batches of export requests.
- This entry contains the number of import requests that were generated (to be processed by the Importer).
- This entry contains the number of items that were exported successfully or that failed during the federation.

Archived

## Useful log entries

Make sure that you look at these log entries:

- **Status of a rule**

  [Scheduler] INFO - Run '50733E7B-02A1-0247-2DD1-FD3B3F2E4926' status is ABORTED

- **Run dispatcher queue state (number of reserved rules/queue capacity)**

  [RunDispatcher] INFO - RunDispatcher wakes up and checks if a run is ready to be processed (6/10)

- **Rule reserved by the run dispatcher:**
  - [RunDispatcher] INFO - Rule 'rule-name' ready for processing
  - [RunDispatcher] INFO - Added work 'F05C44EF-542F-D871-E382-A59893A50879‘

- **Beginning of processing of a rule:**
  - [RunWorker-01] INFO - Start processing RunWork id='F05C44EF-542F-D871-E382-A59893A50879'
  - [RunWorker-01] INFO - Run id=50733E7B-02A1-0247-2DD1-FD3B3F2E4926, criteria= repositoryList=CS, dataMap=CS_DM, contentClass=Customer, queryType=0, maxResults=1

- **Query execution**

  [RunWorker-01] INFO - Executing query : selectionCriteria=(idmId>'003674899') AND (idmId>='003756859'), fullTextCriteria=null, propertyList=[idmId]

- **End of processing of a rule**

  [RunWorker-01] INFO - Finished processing RunWork id='F05C44EF-542F-D871-E382-A59893A50879'

- **Export dispatcher queue state (number of reserved rules/queue capacity)**

  [ExportDispatcher] INFO - ExportDispatcher wakes up and checks if a run is ready to be processed (14/20)

- **Batch (export requests) reserved by export dispatcher:**
  - [ExportDispatcher] INFO - Reserved batch = Batch: batchId=CF10E3B5-B8C6-7289-73A8-501A096B4CE7 runId=1B6D87AD-976D-80AA-43D3-C0B341F4ACFB lockTime=2007-11-27 11:27:51.484 totalItems=1 status=READY
  - [ExportDispatcher] INFO - Added work '6EA6816C-ED5A-6AF0-C8AB-2915CE122F52'

Archived

- **Beginning of processing of a batch:**
  - [ExportWorker-06] INFO - Start processing ExportWork id='6EA6816C-ED5A-6AF0-C8AB-2915CE122F52'
  - [ExportWorker-06] INFO - Processes, batch 'CF10E3B5-B8C6-7289-73A8-501A096B4CE7 of 500 export requests
- **End of processing of a batch**

  [ExportWorker-06] INFO - Finished processing ExportWork id='6EA6816C-ED5A-6AF0-C8AB-2915CE122F52'

## Exporter error handling

The Exporter has a *retry* mechanism, which allows items that failed exporting to be retried several times before being declared as failures. Each error is logged in the fd_error table of the federation database. The number of retries is incremented between each federation until the maximum number of retries (**General Configuration** → **General** → **Max Retries**) is reached for a given item. When an error has been retried the maximum number of times, it is considered a failure.

> **Failure entry management:** A failure entry in the fd_error table remains in the table until the rule is updated or until the failure is removed manually. See “Error management” on page 270 for additional details.

You can obtain detailed descriptions regarding export errors, import errors, or errors resulting in aborting the rule by clicking **FedAdmin** on the Rule management panel.

## Error codes

The error codes are organized in these ranges:

- General: 0-49
- Query phase: 50-99
- Export phase: 200-299
- Import phase: 100-199

Archived

### Common errors

You might see these common errors:

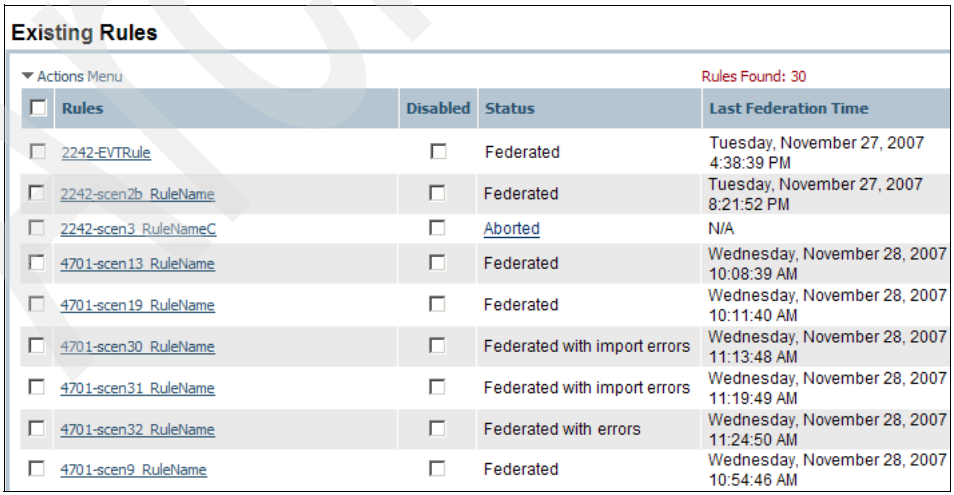
- Unable to connect to the Content Integrator server during the query phase (run worker) or during the export phase (export worker):
  - `51: Unable to retrieve IBM Content Integrator user`
  - `52: Unable to retrieve IBM Content Integrator repository`
  - `230: Unable to retrieve IBM Content Integrator user to process the batch`
  - `240: Unable to retrieve IBM Content Integrator repository to process the batch`
- Unable to retrieve the Content Integrator item during the export phase (export worker):
  - `200: Unable to retrieve IBM Content Integrator item`
  - `210: Unable to export IBM Content Integrator item`

## 5.6.2 View federation rule status

You can view the federation status of a rule by examining the Rule management panel of FedAdmin.

### Rules listing

The Rule management panel, which is shown in Figure 5-52, displays information about the rules that are configured on the system.



**Existing Rules**

Actions Menu — Rules Found: 30

| Rules | Disabled | Status | Last Federation Time |
|---|---|---|---|
| 2242-EVTRule | | Federated | Tuesday, November 27, 2007 4:38:39 PM |
| 2242-scen2b_RuleName | | Federated | Tuesday, November 27, 2007 8:21:52 PM |
| 2242-scen3_RuleNameC | | Aborted | N/A |
| 4701-scen13_RuleName | | Federated | Wednesday, November 28, 2007 10:08:39 AM |
| 4701-scen19_RuleName | | Federated | Wednesday, November 28, 2007 10:11:40 AM |
| 4701-scen30_RuleName | | Federated with import errors | Wednesday, November 28, 2007 11:13:48 AM |
| 4701-scen31_RuleName | | Federated with import errors | Wednesday, November 28, 2007 11:19:49 AM |
| 4701-scen32_RuleName | | Federated with errors | Wednesday, November 28, 2007 11:24:50 AM |
| 4701-scen9_RuleName | | Federated | Wednesday, November 28, 2007 10:54:46 AM |

*Figure 5-52 Rule management panel*

Archived

- The status column can contain the following values:
  - Scheduled
  - Processing
  - Exporting
  - Importing
  - Federated
  - Aborted
- The Last Federation Time is equal to the start time of the previous federation.
- In Figure 5-53, we can see the Progress column, which shows the completion percentage of the current phase:
  - Processing: Query phase
  - Exporting: Export phase
  - Importing: Import phase
- The Items column shows the total number of items that were retrieved by the query. This number is *not* necessarily the number of items that were actually federated.

Rules Found: 30 Show Items: 40

| | Last Federation Time | Progress | Items | Export Errors | Import Errors | Actions |
|---|---|---|---|---|---|---|
| | Tuesday, November 27, 2007 4:38:39 PM | | 1 | | | Reschedule |
| | Tuesday, November 27, 2007 8:21:52 PM | | 1 | | | Reschedule |
| | N/A | | 0 | | | Reschedule |
| | Wednesday, November 28, 2007 10:08:39 AM | | 1 | | | Reschedule |
| | Wednesday, November 28, 2007 10:11:40 AM | | 1 | | | Reschedule |
| import errors | Wednesday, November 28, 2007 11:13:48 AM | | 1 | | 1 | Reschedule |
| import errors | Wednesday, November 28, 2007 11:19:49 AM | | 1 | | 1 | Reschedule |
| errors | Wednesday, November 28, 2007 11:24:50 AM | | 1 | 1 | 1 | Reschedule |
| | Wednesday, November 28, 2007 10:54:46 AM | | 1 | | | Reschedule |

*Figure 5-53 Rule management panel*

Archived

## Errors listing

The errors listing shows the errors that occurred during both the export phase and the import phase of federation. The maximum number of errors displayed is configurable by clicking the following menu options in FedAdmin: **General Configuration** → **General** → **Max Display Errors**.

The Export errors window provides this type of information:

- Errors listed with this icon ( ⚠ ) will be retried.
- Errors listed with this icon ( ❗ ) are considered failures.
- A detailed description is available by clicking this icon ( ℹ ).

Figure 5-54 shows the Export Errors window.

*Figure 5-54 Export errors window*

The Import errors window provides this type of information:

- Import errors are never retried ( ❗ ).
- A detailed description is available by clicking this icon ( ℹ ).

Figure 5-55 shows the Import errors window.

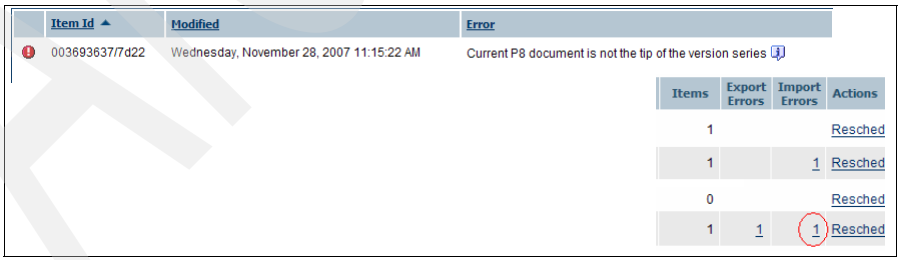



*Figure 5-55 Import errors window*

Archived

## Detailed errors

The Detailed errors panel, Figure 5-56, gives more information about a specific error, including a detailed description, stack trace, and more. You can access the detailed error panel in one of these ways:

- From the rule listing by clicking the Aborted link
- From the errors listing by clicking the information ( ) icon

Unable to process IICE item

```
netica.vbr.client.ItemNotFoundException: COEN0150E: An error occurred while retrieving content or folder (
  at com.venetica.vbr.ejb.bridge.panagon.PanagonBridge.getContent(PanagonBridge.java:1773)
  at com.venetica.vbr.ejb.bridge.panagon.PanagonBridge.getRepoItem(PanagonBridge.java:1835)
  at com.venetica.vbr.ejb.bridge.rmibridge.RMIBridgeProxyImpl.getRepoItem(RMIBridgeProxyImpl.java:235)
  at sun.reflect.GeneratedMethodAccessor6.invoke(Unknown Source)
  at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
  at java.lang.reflect.Method.invoke(Method.java:324)
  at sun.rmi.server.UnicastServerRef.dispatch(UnicastServerRef.java:261)
  at sun.rmi.transport.Transport$1.run(Transport.java:148)
  at java.security.AccessController.doPrivileged(Native Method)
  at sun.rmi.transport.Transport.serviceCall(Transport.java:144)
  at sun.rmi.transport.tcp.TCPTransport.handleMessages(TCPTransport.java:460)
  at sun.rmi.transport.tcp.TCPTransport$ConnectionHandler.run(TCPTransport.java:701)
  at java.lang.Thread.run(Thread.java:534)
  at sun.rmi.transport.StreamRemoteCall.exceptionReceivedFromServer(StreamRemoteCall.java:247)
  at sun.rmi.transport.StreamRemoteCall.executeCall(StreamRemoteCall.java:223)
  at sun.rmi.server.UnicastRef.invoke(UnicastRef.java:133)
  at com.venetica.vbr.ejb.bridge.rmibridge.RMIBridgeProxyImpl_Stub.getRepoItem(RMIBridgeProxyImpl_Stub.jav
  at com.venetica.vbr.ejb.bridge.rmibridge.RMIBridge.getRepoItem(Unknown Source)
  at com.venetica.vbr.ejb.access.AccessServicesImpl.getRepoItem(Unknown Source)
  at com.venetica.vbr.client.Content.getVersion(Unknown Source)
  at com.filenet.cfs.iice.federator.worker.ExportWorker.processWork(ExportWorker.java:223)
  at com.filenet.cfs.iice.federator.worker.Worker.run(Worker.java:57)
 by:
tionException: 0x80043603 -
  at com.linar.jintegra.NativeObjRef.nativeVtblInvoke(Native Method)
  at com.linar.jintegra.NativeObjRef.a(Unknown Source)
  at com.linar.jintegra.Dispatch.vtblInvoke(Unknown Source)
```

*Figure 5-56 Detailed errors window*

## Error management

From the Export and Import error list, you can take the following actions if the rule is in a FEDERATED or ABORTED state:

- Delete:
  - Delete the selected errors and failures.
  - Selected errors and failures are removed from the database and cannot be recovered.
- Delete All:
  - Delete all errors and failures.
  - All errors and failures are removed from the database and cannot be recovered.
- Ignore:

  - Ignore selected errors by converting them into failures.
  - The retry count of the selected errors is set to the maximum number of retries allowed.
- Ignore All:
  - Ignore all errors by converting them into failures.
  - The retry count of all errors is set to the maximum number of retries allowed.
- Reset:
  - Reset selected failures by converting them back into errors.
  - The retry count of the selected errors is reset to 0.
- Reset All:
  - Reset all failures by converting them back into errors.
  - The retry count of all errors is reset to 0.
- Retry:

  Try the selected export or import errors again. A partial federation is launched, meaning that the query phase is skipped and only errors are processed.

### 5.6.3 Installation and Configuration problems

We list several common mistakes that might occur during installation and configuration.

#### Database-related problems

You might encounter these situations:

- The database schema might not be configured correctly.
- Make sure that you run the **`FedDatabaseScript.sql`** on the master database, as well as on each federator database, to create the exporter tables.
- Verify that when you create a new FCD, the importer tables are created properly.

Here is a list of tables to consult if you run into issues.

Consult these Exporter tables:

- fd_XML_file
- fd_credential
- fd_exception

- fd_run
- fd_batch
- fd_exp_queue
- fd_error

Consult these Importer tables:

- cfs_imp_session
- cfs_imp_queue
- cfs_imp_queue_error
- fdb_version_series

## JDBC Drivers

You might encounter these situations:

- If the following exception is found: "`java.sql.SQLException: No suitable driver`", verify that the JDBC driver is installed and configured.
- FedAdmin:
  - (WebSphere):
    - The JDBC library needs to be added to the classpath of the application server.
    - A jdbc.drivers Java property has to be set for the application server.
  - (WebLogic):
    - The JDBC library needs to be added to the classpath in the startup script.
    - A jdbc.drivers Java property has to be set in the startup script.
- Exporter:

  The JDBC library must be located in the *CFS_HOME*`\FedExporter\lib` directory.

Table 5-5 lists the various JDBC libraries and the related class names.

*Table 5-5 JDBC libraries*

| Product | JDBC library | Class name (jdbc.drivers property) |
|---|---|---|
| DB2 | db2jcc.jar and db2jcc_license_cu.jar | com.ibm.db2.jcc.DB2Driver |
| MS SQL Server 2005 | sqljdbc.jar | com.microsoft.sqlserver.jdbc.SQL ServerDriver |
| Oracle | ojdbc.jar | oracle.jdbc.driver.OracleDriver |

Archived

## Additional considerations

Also, consider these areas:

- Session pool issues:
  - Unable to locate the session pool named SessionPool-cssql1.
  - Make sure that you created a session pool for each external repository with the following naming convention: SessionPool-*REPOSITORY_NAME*.
- Clock synchronization is important.

  The clocks of all servers (CFS/IICI, DB, P8, and CS) must be synchronized for the sweeping mechanism to work properly.
- Deadlock errors occur when federating large quantities of documents:
  - Cause: When federating large quantities of documents, you might receive the following error, as shown in Example 5-6.

    *Example 5-6 Deadlock example*

    ```
    ERROR - Unable to retrieve batch
    id='E8B1739C-FF4B-8B4B-9150-82BB21B54FA5'
    com.filenet.cfs.iice.federator.exception.DataAccessException:
    com.microsoft.sqlserver.jdbc.SQLServerException: Transaction
    (Process ID 59) was
    deadlocked on lock resources with another process and has been
    chosen as the deadlock
    victim. Rerun the transaction.
    Caused by:
    com.microsoft.sqlserver.jdbc.SQLServerException: Transaction
    (Process ID 59) was
    deadlocked on lock resources with another process and has been
    chosen as the deadlock
    victim. Rerun the transaction.
    ```

  - Solutions:
    - Use a smaller Java virtual machine (JVM) heap size for the RMI Bridge. A smaller JVM heap size forces more frequent JVM garbage collection and, depending on the implementation of the RMI bridge, might allow locked resources to be released at a faster rate, hence avoiding deadlock.
    - Reduce the batch size value using the CFS Administrator tool.
- Mime type for a federated document is incorrect.

  Solution: Create a new version of the document in the external repository with the correct mime type. The next time that the rule is run, the updated

Archived

document will be detected, and the mime type of the new version will be applied to the federated document.

- An import error is not cleared when the rule is rescheduled:
  - Cause: If there is an import error (such as due to a federated item being versioned), this error is logged, as expected. However, when the rule is rescheduled, the error is not cleared.

    If there is an import error, the Rule management window in FedAdmin indicates that an error occurred. If you modify the import rule as part of the process of resolving the import error, the error is cleared from the Rule management window.

    If the error is resolved without editing the rule (for example, by adding or modifying a property in Content Engine), the import rule has not been changed. Rerunning the same import rule might succeed, because the problem was fixed, but the error is still indicated in the Rule management window.
  - Solution:
    - In the Rule management window, click **Edit** to edit the rule with the error.
    - Click **Apply**. You do not need to make any changes.
    - Rerun the rule.
- Import fails when attempting to create a second version of a federated document:
  - Cause: If a Content Engine document property is defined as both “required” and “not copied to reservation”, the Importer fails when it attempts to create a second version of a federated document, because the document checkout fails. The request remains in the queue and is retried indefinitely.
  - Solution: Change the property definition to be either not required or make it copied to reservation. After the change is made, the next time that the Importer retries the request, the newly federated version will be created.
- E_OBJECT_MODIFIED error

  An object modified error might be generated when an attempt is made to import a new document with multiple versions and to file it in a folder. The error can be ignored, because the request will be retried automatically.

Archived

## 5.7 Federated records management with CFS (using Content Integrator)

You can use CFS implementation (using Content Integrator) to federate the following repositories:

- IBM Content Manager
- IBM FileNet Content Manager
- IBM FileNet Content Services
- Documentum Content Server
- Open Text Livelink Enterprise Server

In this section, we focus on IBM Content Manager (CM8) as an example of working with CFS (using Content Integrator). The federation process and functionality might vary slightly among the supported repositories.

### 5.7.1 Lockdown behavior

You can configure the IBM CM8 connector to enable Content Integrator to lock and unlock the documents that are stored in a CM8 repository. The action of federating a document from CM8, in itself, does not lock down the content. To enable CFS implementation (using Content Integrator) to lock and unlock documents that are stored in a CM8 repository, you must enable CM8 for records management through the Enterprise Records (formerly known as IBM FileNet Records Manager). After CM8 is enabled for records management, any user with the ItemRecordsAdmin privilege can lock and unlock the content. Users who have the AllPriv privilege set, such as icmadmin, can lock and unlock documents.

You can lock and unlock documents through the lock down record method and undeclare a record through the unlock record method. The lockDownRecord method declares the document under records management, and the unLockRecord method undeclares the document from the records management.

When a document is locked, all users, including the icmadmin super user, cannot check in the native content. Any user with the AllPriv privilege set can unlock the content even if it was locked by the super user. Locking and unlocking a document is enabled at the content object-level only and is supported only by CM8.

You must perform the following tasks for lockdown:

- CM8 must be records enabled.
- CFS must use an account with adequate permissions to lock down or delete content.

- The Usage Mode property of the Content Integrator CM8 connector must be set to  Records Management.
- CM8 item versioning policies must match the policies that are described in “Records Management mode” on page 279.
- If attributes between the federated document (in the IBM FileNet content repository) and the source document (in IBM Content Manager) differ, the lockdown will fail. If the lockdown fails, the records declaration will also fail and will log an event in the IBM FileNet error log.

CFS first checks to see that it can lock down the content. After CFS verifies that it can lock down the content, it sets the flag in the ICMManagedRecord to indicate to the system that the document must be locked.

After a document is declared as a record, the content and metadata for that record remain in the CM8 repository. Only an authorized user with ItemRecordsAdmin privileges, such as the records administrator, can process or manage the life cycle of the record. Users that do not have the ItemRecordsAdmin privilege are prevented from performing the following operations on an Item that has been declared a record:

- Delete the item.
- Update attributes associated with the item.
- Update the content that is associated with the item (document parts).
- Reindex the item.
- Create a new version that causes the number of extent versions to exceed the maximum number that is allowed by the Item Type definition.

**Performance consideration:** Always take into consideration the extra load that is added to the existing system when working with records. For instance, the lockdown operation, when performed on a large number of federated documents, can consume a significant amount of system resources, which need to be accounted for. Records Administrators performing a high volume of searches for records can also be costly. The Enterprise Records sweep processes (disposition and hold) can also greatly increase the system load. Plan ahead to insure that you have sufficient hardware to handle your operations and that you have sufficient time to complete these operations.

## 5.7.2 Deletion behavior

When a record reaches its end of life and you (or the system) initiate the record deletion process, Enterprise Records sends a request to CFS to reset the ICMManagedRecord flag and to delete the document from CM8. Enterprise

Records then proceeds to delete the record and the federated document in the IBM FileNet content repository.

In order to complete these operations, the CFS user must have adequate permissions, which is typically addressed by membership to the appropriate Records Administrator or Records Manager group in Enterprise Records.

### 5.7.3 Configuring and enabling Enterprise Records

The manner in which CM8 is configured can affect how CFS and Enterprise Records function. You must configure CM8 appropriately in order for CFS to be able to federate, lock down, and delete content.

In order for CFS to lock down content, CM8 must be enabled to support P8 Records Management. Follow these steps:

1. Expand the library node in the left pane of the CM System Administration Client, as shown in Figure 5-57.
2. Expand **Library Server Parameters** in the left pane.
3. Click **Configurations** in the left pane.
4. Right-click the **Library Server Configuration** icon in the right pane. Select **Properties** from the context menu, which invokes the Properties dialog. Select the **Features** tab.

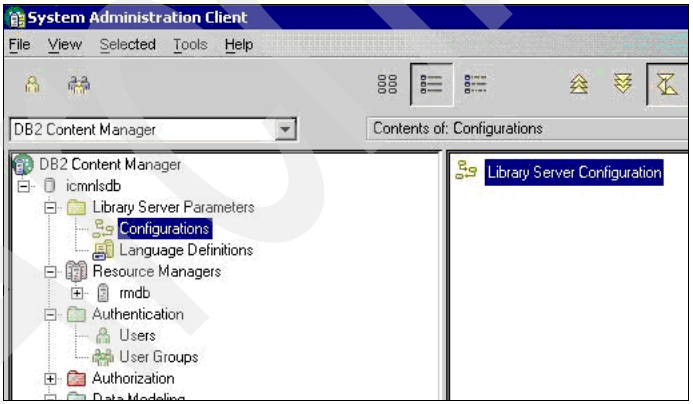



*Figure 5-57 Configure records management on CM8*

5. In the Library Server Configuration window (Figure 5-58 on page 278), select **Enable Records Management**.
6. Select **P8 Records Manager**.

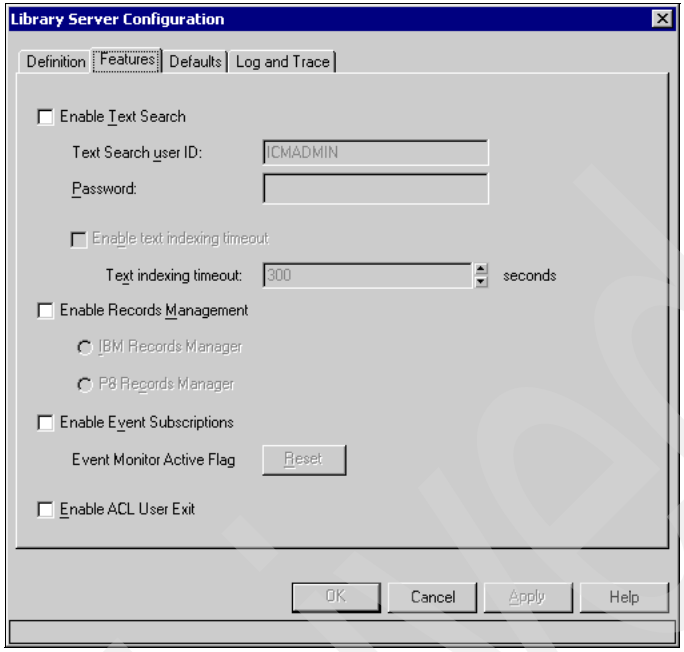



*Figure 5-58 Enable records management on CM8*

Enabling records management in CM8 allows the ICMManagedRecord attribute to be set. When an item is federated and brought under records management using Enterprise Records, this attribute is set on the referenced item, which prevents changes or deletions.

If the object type is set to allow versioning on updates, the CFS property update will fail. In this instance, the act of updating the property in order to lock down the content creates a new version, which in turn needs to be locked down, in essence creating a loop. Therefore, the versioning of all metadata classification types, which include item, resource item, and document, must be set to `Version - Never`. The only classification type that can be versioned is document part, which must be set to `Version - Always`. This configuration allows the content to be versioned without negatively affecting the ability for CFS to lock down content when appropriate.

## Configuring the Usage Mode

When using CFS implementation (using Content Integrator), you configure the Content Integrator CM8 connector in either records management mode or search mode. If you use Enterprise Records, you must configure the CM8 connector in records management mode.

In search mode, CM8 content that can change due to its versioning policy is not federated to P8, which protects the ability to search on keywords contained in source documents that do not change.

The usage mode is configured using the Content Integrator Administration Tool to set a custom property on the IBM CM8 connector. The custom property is named UsageMode and must be assigned one of two values:

- Search and Retrieve
- Records Management

Use this syntax: UsageMode=<*value*>:

- The name, UsageMode, is case sensitive, but the value is not.
- The value does not need to be enclosed in quotation marks even though it includes spaces.

### *Records Management mode*

When the Usage Mode custom property is set to *Records Management*, the following conditions apply:

- CM8 Items that have a new version policy for attributes set to Always create are not federated.
- CM8 Items that have a new version policy for their document parts set to Never create or Prompt to create are immediately locked down upon federation.
- CM8 Resource Items that have a new version policy for attributes set to Never create or Prompt to create are immediately locked down upon federation.
- CM8 Resource Items that have a new version policy for attributes set to Always create are not federated.

### *Search and Retrieval Mode*

Search and Retrieval mode must be set when CFS is not used with Enterprise Records. When this mode is set, the following conditions apply:

- CM8 Items that have a new version policy for any of the document parts set to Never create or Prompt to create are not federated.
- CM8 Resource Items that have a new version policy for attributes set to Never create or Prompt to create are not federated.

**Note:** Both the new version policy for attributes and the new version policy for document parts are specified as part of the Item Type definition.

Archived

For best practices related to records-managed federated documents, see 5.8.5, “Records management-related best practices” on page 287.

## 5.8 Best practices

In this section, we discuss recommendations about how best to configure aspects of your CFS implementation (using Content Integrator) system.

### 5.8.1 Content Integrator RMI proxy connector performance considerations

In this section, we discuss performance considerations for the Content Integrator RMI proxy connectors.

#### Source repository RMI proxy connector

We recommend that you run the source repository RMI proxy connector as physically close to the source repository as possible. If possible, run the RMI proxy connector on the same machine as the source repository. However, if you cannot run the RMI proxy connector on the same machine as the source repository, or if that puts too much load on the source repository, run the connector on a machine within the same local area network (LAN) as the source repository.

The source repository connector is used heavily during the exporting phase of CFS implementation (using Content Integrator) by the CFS exporter. It is also used during the retrieval of federated content by P8 applications. So, try to minimize the time periods when this connector is used concurrently by both of these separate activities. For example, you can schedule the CFS exporter rules to run only in the evening hours when there might be fewer users retrieving content using P8 applications. Thus, during the daytime hours, user retrieval performance will not be impacted by any export processing.

We also recommend that you run at least two RMI proxy connector instances for your source repository. If your source repository is scaled across multiple host machines, we recommend running at least one RMI proxy connector for each host.

Archived

### Destination repository RMI proxy connector

The destination repository RMI proxy connector is your P8 Content Manager connector to which you are federating documents. The destination P8 connector is only used when you create federation rules. It is *not* used by the CFS runtime components, such as the exporter, importer, and content retrieval.

Because FedAdmin is the only component that uses the destination P8 connector, we recommend that you run this connector on the same server where you have deployed FedAdmin. The P8 connector does not use many computing resources. So, we recommend that you leave the P8 connector running all the time. However, you can shut it down, if you want, after you have finished defining all the federation rules that you plan to use.

## 5.8.2 Exporter performance considerations

In this section, we discuss performance considerations for the CFS exporter.

### General exporter optimization

Consider these potential effects on performance:

- If the processing of each item takes too long, reduce the batch size to avoid having export workers “lease” batches for a long period of time. You can configure reducing the batch size by selecting **General Configuration** → **Run Workers** → **Batch Size**.
- You can adjust the number of threads to take advantage of the hardware running the application. Configure adjusting the number of threads by selecting **General Configuration** → **Run/Export Workers** → **Max Number of Workers**.
- On environments that run slowly, we recommend that you increase the lease time for runs and requests. Configure this increase by selecting **General Configuration** → **Run/Export Workers** → **Run/Request Lease**. The *lease time* is the amount of time that is given to a run worker (export worker) to process a rule (a batch) before the batch is considered abandoned by the worker so that another worker can pick up the batch.

### Query phase optimization

Consider these potential effects on queries:

- Use the *chunking* mechanism to process rules that federate a large number of items. The chunk size must be adjusted based on each connector (external repository) limitation. Configure the chunk size by selecting **General Configuration** → **Federation Query** → **Chunk Size**.

Archived

- Use the *sweeping* mechanism to optimize and minimize the work that is generated for each federation.
- You can obtain detailed information about *chunking* and *sweeping* in the online FileNet P8 documentation by selecting **System Administration** → **Content Engine Administration** → **Content federation** → **CFS-IBM Content Integrator** → **Configuring chunking and sweeping**.

### Changing the CFS administrator user password

Use the FedAdmin Advanced Configuration tab's Application Password field to change the CFS administrator user password, which is the user account that you use to log in to FedAdmin.

### Changing the federation database password

To change the federation database password, perform these steps:

1. Use the database-specific procedure to change the password of the user account that you use to access the federation database.
2. If the federation database is the *master* federation database, as described in "Advanced Configuration" on page 226, use the CFS security application that is located in the `CFS_HOME\FedExporter\bin` directory, as shown, to change the password:

   ```
   security changeDBPassword <new password>
   ```

3. For other federation databases, that is, *not* the master federation database, use the FedAdmin Advanced Configuration tab's Federator Databases section to change the database password.

### Running multiple exporter instances

You can run multiple exporter instances on separate hosts to improve performance. Copy the `fed_exporter.zip` file from your `CFS_HOME` directory and extract it on another machine where you want to run another exporter instance.

This zip file is automatically created when you first install CFS. It contains the contents of the `CFS_HOME\FedExporter` directory, plus a copy of the Java installation on your current machine. When you extract it on another host, you can start the CFS exporter by executing the **`run_exporter`** script file in the extracted `bin` directory.

If you change either your CFS administrator user password for FedAdmin or the master federation database password, you must regenerate this zip file on your *original* CFS installation machine. Then, you must copy and extract this zip file over your other CFS exporter directories on the other hosts. To recreate the zip file, execute the `CFS_HOME/create_fed_exporter_zip` file on your *original* CFS installation host.

### Tips for FedAdmin Rule Builder

Follow these tips:

- Select a **Data map** first to display the list of available Document classes that are selectable for the criteria.
- You can only use the Full-Text Criteria field if the source repository supports it.
- For the Property Criteria field, make sure to select a supported operator based on the selected property. For example, a DATE property does not support the LIKE operator.
- Test the rule criteria using **Search** with Max Results set to `1`, if the rule is supposed to return a large number of items.
- Remember to remove the Max Results value before scheduling your rule to be run by the CFS exporter. Otherwise, you can inadvertently decrease the efficiency of the federation process, as described in 5.5.4, “Create federation rules” on page 247.

### Tips for FedAdmin rule management

Follow these tips:

- Use the **Launch** action in the Rule Listing to immediately execute a rule, regardless of the configuration in the Service Scheduler.
- Always set up a default schedule.

## 5.8.3 Importer performance considerations

Because the importer runs as part of P8 CM, improving the performance of P8 CM can improve the performance of the importer. You can obtain general tips for increasing the performance of P8 CM in the *IBM FileNet P8 Performance Tuning Guide*:

`http://www.ibm.com/support/docview.wss?rs=3278&uid=swg27010422`

### Increasing Importer throughput

To increase the importer rate on the P8 CM server, consider increasing the Importer Threads parameter and decreasing the Dispatcher Wait Time Interval parameter. You access these properties from IBM FileNet Enterprise Manager by selecting ***domain*** → **Properties** → **CFS Import Agent** tab. Adjust these values to optimize your system resources.

Archived

### 5.8.4 Configuring CFS implementation (using Content Integrator) for a highly available environment

In this section, we describe the installation of CFS implementation (using Content Integrator) in a supported FileNet P8 High Availability (HA) environment. We cover the CFS components only. For details about configuring other P8 applications in a highly available environment, see the *IBM FileNet P8 High Availability Technical Notice*:

http://www.ibm.com/support/docview.wss?rs=3278&uid=swg27010422

In an HA CFS implementation (using Content Integrator) environment, the CFS FedAdmin, CFS exporter, and Content Integrator components run on each of the HA nodes designated for CFS implementation (using Content Integrator). Each of the installations on each node must be duplicates of each other. They must also be installed in the *same* directory locations on each node. In general, make configuration changes to CFS federation rules and data mapping on a single node. Then, copy the changed configuration files to each of the other nodes in your HA environment.

Use the following steps to configure CFS implementation (using Content Integrator) for a HA environment:

1. Run the Content Integrator installer on all of the HA nodes that you have designated for CFS implementation (using Content Integrator). You *must* install Content Integrator in the *same* directory location on all HA nodes.
2. Perform *all* of the Content Integrator configuration on only *one* node in your HA environment:
   a. Run the same RMI proxy connectors on each of your HA nodes. For example, if you run a P8 connector on one HA node, you must run that same P8 connector on all HA nodes.
   b. Make sure that you copy all required client libraries that are used by each of your connectors to all of your HA nodes. For example, for the P8 connector, copy the `ICI_HOME\lib\Jace.jar` library file and the `ICI_HOME\lib\javaapi.jar` library file to the same location on each HA node. If the P8 server runs on WebSphere, make sure that you also install the WebSphere Application Client on each HA node.
   c. When configuring the RMI proxy connector URLs in the Content Integrator administration tool, add a URL for each of the connectors that runs on a HA node. For example, if you run a P8 connector on HA nodes cfsiciserver1 and cfsiciserver2, your RMI proxy connector URL configuration looks like these lines:

      ```
      rmi://cfsiciserver1:1251/RMIBridgeServer
      rmi://cfsiciserver2:1251/RMIBridgeServer
      ```

d. Create the CFS FedAdmin user following the instructions in 5.4.5, “Create the CFS federation administrator user” on page 224.

3. Save your configuration changes in the Content Integrator administration tool, and then, copy the following files that are located in the *ICI_HOME* directory to the same location on all HA nodes:

   – `config.xml`
   – `sso.xml`

4. Run the CFS installer on all HA nodes that you have designated for CFS implementation (using Content Integrator). You *must* install CFS in the *same* directory location on all HA nodes:

   a. When prompted for the Content Integrator single sign-on (SSO) Server URL, use `localhost` instead of your server host name or IP address, as shown the following example:

      ```
      rmi://localhost:1250/SSOServer
      ```

   b. When prompted for the Content Integrator Configuration Server URL, use `localhost` instead of your server host name or IP address, as shown the following example:

      ```
      rmi://localhost:1250/ConfigurationServer
      ```

   This setting forces the CFS exporter and FedAdmin running on the same host to use the Content Integrator services that are also running on that same host instead of trying to use Content Integrator services running on a separate host.

5. The CFS installer generates a master encryption key that is used to encrypt the CFS admin user password and the master federation database password. This encryption key and the encrypted passwords are stored in the *CFS_HOME*`\FedExporter\conf` directory. This `conf` directory *must* be identical for the CFS installations on all HA nodes. So, after running the CFS installer on all HA nodes, select *one* of your CFS nodes, and copy its *CFS_HOME*`\FedExporter\conf` directory to the same location on all of the other CFS HA nodes.

6. Deploy FedAdmin to each HA node.

   Run an application server on each of the nodes in your HA environment, and deploy the FedAdmin Web application to each of them. The deployment of FedAdmin varies depending on the application server type that you choose to use. Refer to the application server help for deploying Web applications. You can use the clustering capability of the various application servers or deploy FedAdmin to a stand-alone instance of the application server on each node.

Archived

**Note:** The FedAdmin war file contains a property file that specifies the locations of *ICI_HOME* and *CFS_HOME* on the *local* machine. Because of this property file, you can only deploy FedAdmin on a host where both Content Integrator and CFS have been installed. Also, because of this property file, the installation locations of both Content Integrator and CFS must be the same across all nodes in an HA environment.

7. Configure P8 to connect to a HA CFS environment.

   Use IBM FileNet Enterprise Manager to configure the Content Integrator fixed content device to reference a HA Content Integrator environment. When specifying the Content Integration URL, use a comma-separated list to each of the Content Integrator servers, as shown, where cfsiciserver1 and cfsiciserver2 are your HA Content Integrator nodes:

   ```
   rmi://cfsiciserver1:1250/ConfigurationServer,rmi://cfsiciserver2:125
   0/ConfigurationServer
   ```

8. Replicate Content Integrator configuration changes.

   When you use the Content Integrator Administration Tool to make configuration changes to any of the Content Integrator connectors or data mappings, you must only do so on one of the nodes in your HA environment. Then, follow this procedure:

   a. Save your configuration, and exit the Content Integrator Administration Tool.

   b. Copy the *ICI_HOME*`\config.xml` file to the same location on each of your Content Integrator HA nodes.

   c. Restart the VeniceBridgeServices, RMI bridges, and CFS exporter processes that run on all of the nodes in your HA environment.

9. Replicate password changes.

   If you change either your CFS administrator user password, as described in “Changing the CFS administrator user password” on page 282, or the master federation database password, as described in “Changing the federation database password” on page 282, perform the following steps from the same node where you made these password changes:

   a. Copy the *ICI_HOME*`\sso.xml` file to the same location on each of your other CFS HA nodes.

   b. Copy all of the files under *CFS_HOME*`\FedExporter\conf` to the same location on each of your other CFS HA nodes.

   c. Restart all of the FedAdmin application servers and CFS exporters in your HA environment.

### 5.8.5 Records management-related best practices

When using records management on federated IBM Content Manager documents, we recommend the following best practices:

- Avoid setting a maximum version limit on item type definitions.
- Avoid setting the retention period within IBM Content Manager.
- Use document item types rather than resource item types whenever possible.
- Avoid using reference attributes.

Archived

Archived

# Part 3

# IBM Content Integrator

Archived

© Copyright IBM Corp. 2010. All rights reserved.

Archived

6

# Content Integrator architecture

In this chapter, we describe the logical architectural makeup of IBM Content Integrator Version 8.5.1. We introduce the three architectural services layers that make up Content Integrator. In addition, we provide a detailed description of the key components that constitute each of these services layers. We also address additional services that are provided with the product, as well as a collection of overall design principles followed in the development of the product.

We cover the following topics in this chapter:

- Architecture overview
- Integration Services layer
- Federation Services layer
- Developer and User Services layer
- Additional services
- General design principles
- Recent product enhancements

© Copyright IBM Corp. 2010. All rights reserved.

## 6.1 Architecture overview

Implemented using Java, the architecture of Content Integrator can be loosely categorized into the three functional areas that are presented in Figure 6-1: integration services, federation services, and developer and user services.

In general terms, the components that make up the Integration Services layer interact directly with underlying enterprise repository systems to provide consistent and efficient access to content. Building on integration services, the Federation Services layer provides access to content from multiple repositories in a single search request. Finally, the Developer and User Services layer provides the set of application programming interfaces (APIs) made available to take advantage of the capabilities provided by Content Integrator.

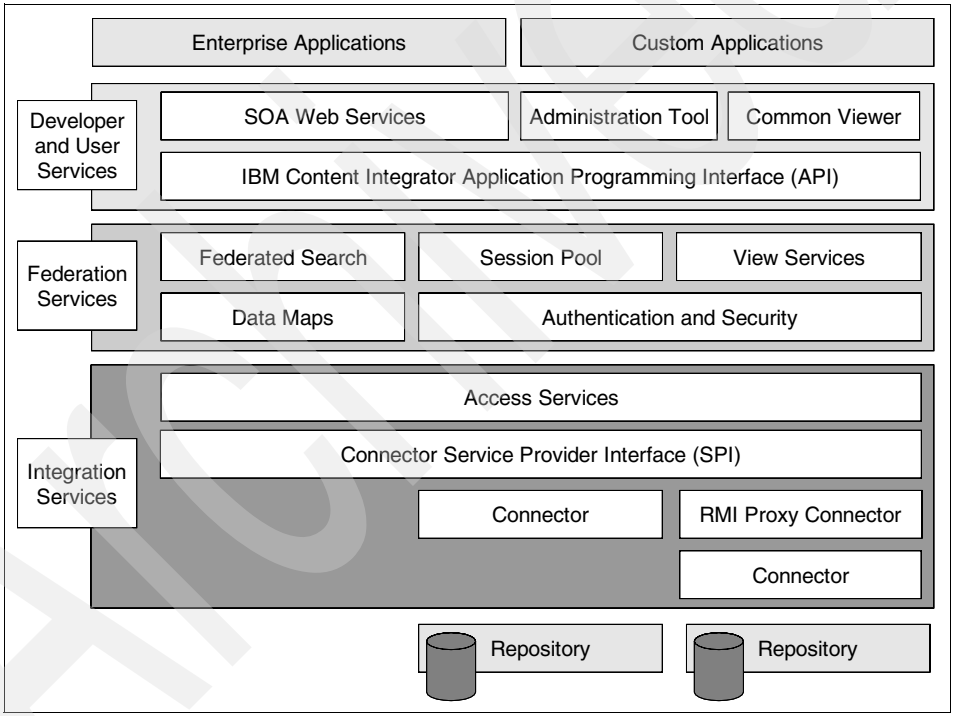



*Figure 6-1 Content Integrator general architecture*

Each of these architectural layers, as well as several supporting services that are provided by Content Integrator, is detailed in the remainder of this chapter.

## 6.2 Integration Services layer

Key components making up the *Integration Services layer* communicate directly with native repository systems to provide consistent and efficient access to enterprise content. Using the connector service provider interface (SPI), you can build connectors to expose common content management activities, such as the creation, update, and retrieval of content, searching and browsing for content, as well as more advanced features of content annotation and content security.

There are hundreds of content management systems in use today, with a wide variety of functional capability that is provided by each system. Making use of the *repository profile*, an implementation detail that is provided with the connector SPI, each repository's ability can be determined programmatically. This capability allows Content Integrator clients the flexibility to take advantage of the full range of services provided by specific content management systems. The only limitations are those of the native content management system.

Details about the key components that make up the Integration Services layer follow.

### 6.2.1 Connectors and the Connector SPI

Considered one of the most intricate components in all of Content Integrator, the *connector* is responsible for translating client requests into repository-specific API calls. Consistency in connector implementations built across all enterprise content management systems is maintained using the Connector SPI. The Connector SPI allows applications developed against Content Integrator to remain repository-agnostic; you can add connectors for any number of disparate content management systems seamlessly.

Using Figure 6-2 on page 294 as a guide, you can follow the general thread flow through the Integration Services layer from its origin in either the Developer and User Services layer or the Federation Services layer to the content management system's repository. Next, the access services component decides through which connector to route the request. After the request reaches the appropriate connector, a translation is made from the Content Integrator constructs to those constructs required by the content management system's API. If any content data or metadata was requested, it is returned to the connector where a translation into Content Integrator constructs is performed before being returned back to the client.

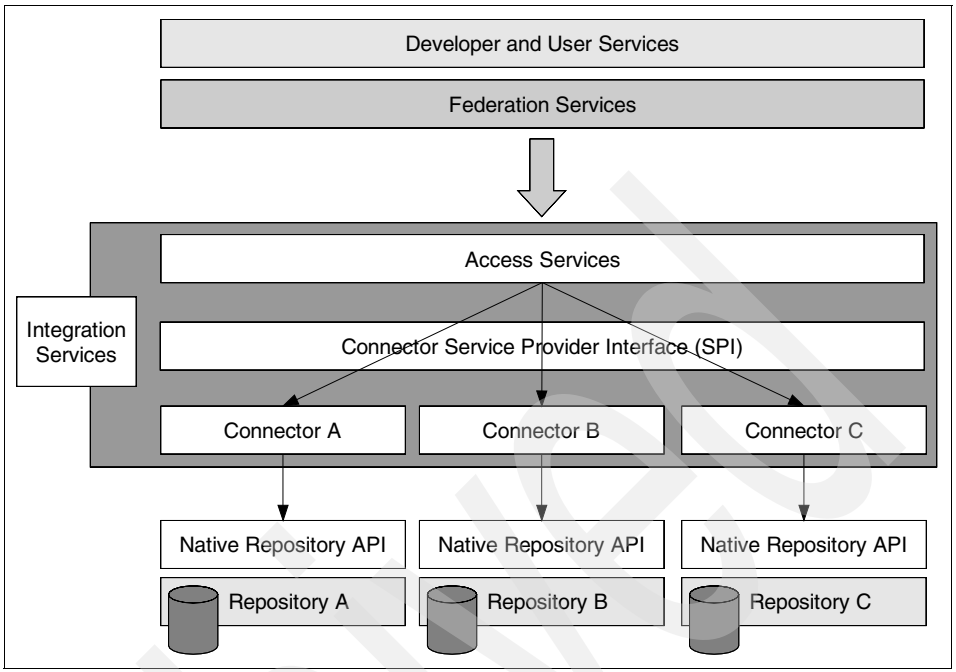



*Figure 6-2 General thread flow through the Integration Services layer*

Using the Connector SPI, you can incorporate the full range of content management system capability into a connector implementation. The various categories of actions that are supported include but are not limited to these actions:

- Logon/logoff
- Retrieve options
- Create options
- Update options
- Delete options
- Query and browsing options
- Security and records management options
- Check-in/check-out

When building custom connectors, it might only be necessary to provide support for logon/logoff, a query option, and a simple retrieve option. For many application scenarios, such as an extract-transform-load or a Web crawling use case, this type of read-only connector implementation might be all that is necessary. Of course, supporting all of the actions making up the Connector SPI makes for an extremely robust connector implementation.

Archived

**Connector development:** It is important to understand that connector development efforts do not require that the content management system's API be available in Java. Successful connector implementations are certainly possible using Component Object Model (COM)-based APIs, Web services-based APIs, and any number of other technologies that you can bridge to Java.

Content Integrator Version 8.5.1 provides the following connector implementations:

- EMC Documentum
- File system
- Hummingbird Document Management (DM)
- IBM Content Manager
- IBM Content Manager OnDemand
- IBM FileNet Business Process Manager (BPM)
- IBM FileNet Content Services
- IBM FileNet Image Services (IS)
- IBM FileNet Image Services Resource Adapter (ISRA)
- IBM FileNet Content Manager
- IBM Lotus® Domino® Document Manager
- IBM Lotus Notes®
- IBM WebSphere MQ Workflow
- IBM WebSphere Portal Document Manager
- Interwoven TeamSite
- Microsoft NT File System
- Microsoft Windows SharePoint Services
- Open Text Livelink
- Relational database management system (RDBMS)

You can see the supported repository versions at this Web site:

http://www.ibm.com/support/docview.wss?rs=86&uid=swg27015930

### 6.2.2 Access services

While connectors are considered the most intricate component in the Integration Services layer, access services is respected as the architectural heart of all of Content Integrator. As you can see in Figure 6-2 on page 294, the *access services* component is responsible for synchronizing and routing all client requests to their appropriate connector deployments. As the communication center, result set consolidation from disparate content management repositories is handled when federated searches are requested by client applications. Finally,

Archived

it is through the access services component that configuration settings and updates are gathered and dispersed across the system.

### 6.2.3 Remote Method Invocation (RMI) Proxy Connector

Less of a true component and more of a runtime service, the *Remote Method Invocation (RMI) Proxy Connector* allows connector implementations to be deployed in their own Java virtual machine (JVM). Using the Java RMI technology, Content Integrator object instances in one JVM can communicate with connector object instances in another JVM. Furthermore, the separate JVMs do not need to be colocated in the same operating system environment. This distinction allows for a number of deployment options that can take advantage of any number of computing resources that are made available.

For information about the RMI proxy connector service and example deployment options, see 6.5.3, “RMI proxy connector server” on page 309.

## 6.3 Federation Services layer

Direct interaction with content management systems is the responsibility of the Integration Services layer, and in many business contexts, this layer is often sufficient to meet client needs. Building upon integration services, the *Federation Services layer* adds a collection of convenience and efficiency components that you can easily incorporate into applications requiring wider control across an enterprise environment. For example, you can utilize federated search to locate content residing in disparate repositories using a single search request.

These key components make up the Federation Services layer:

- Federated search
- Data maps
- Session pool
- Authentication and security
- View services

We describe each component in detail in the following sections.

### 6.3.1 Federated search

Likely considered one of the most important capabilities of an enterprise content management system is the ability to perform searches for persisted content. A request as simple as “retrieve all activity for customer XYZ since January 2000” can likely provide a customer service representative with vast amounts of

information, assuming that it is all contained in a single content management system. However, if the data is spread out over disparate systems through mergers and acquisitions over time, or even through natural organic growth of the business internally, this seemingly simple request can actually result in a series of tasks, burdening the customer service representative who is in search of needed information. Content Integrator's federated search was developed specifically to handle this scenario.

*Federated search*, also referred to as a federated query or a multi-query, is a key federation services component that manages dispatching real-time search requests to disparate content management systems specified to participate in such a request. Specifically, you can initiate a single federated query to search all content management systems making up an enterprise environment. The results of the search request are then aggregated and returned back to the requesting client. Using federated search, the customer service representative in our previous example is able to gather all of the necessary information in a simple request.

A general thread flow for a federated search request is presented in Figure 6-3 on page 298. In this example, a federated search request has been received from the Developer and User Services layer:

1. The federated search component determines that the search requests participation from all three content management systems: Repository A, Repository B, and Repository C.
2. Using the Integration Services layer, the federated search component dispatches asynchronous search requests for each content management system.
3. Integration services handles each query and returns the results back to the federated search component.
4. The results from the three content management systems are then aggregated into a single result set.
5. Post-process sorting is performed on the results if requested by the original federated search request.
6. The results of the federated search request are returned to the Developer and User Services layer.

Archived

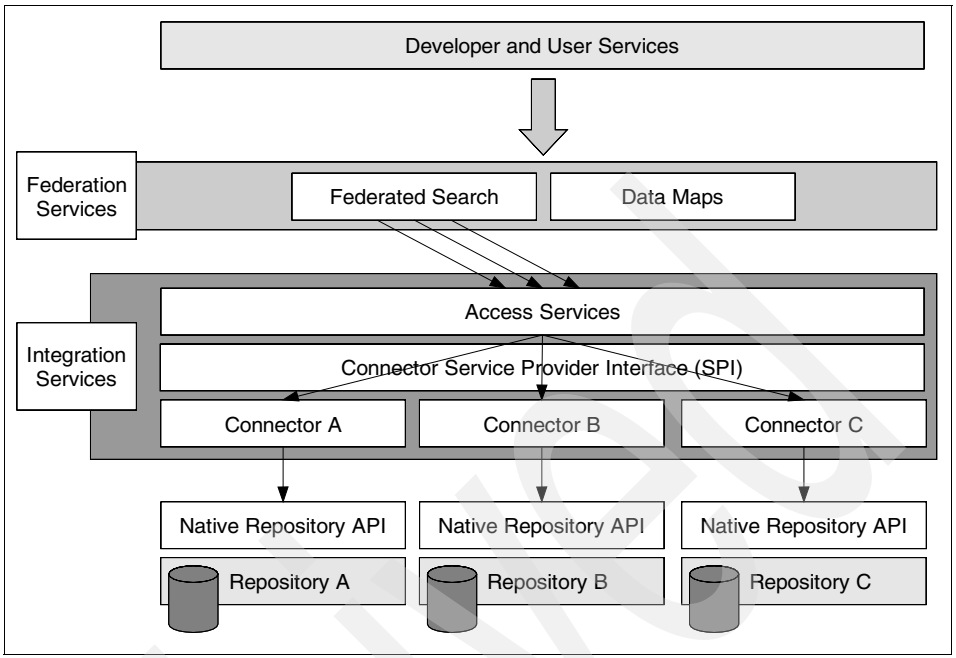



*Figure 6-3 General thread flow for a federated search request*

## 6.3.2 Data maps

*Data maps* are used to designate a mapping between repository-specific property names and common, application-oriented names. These common names, called *data map elements*, are typically used to replace repository properties having vague or cryptic identifiers with names that are much easier to recall and understand. Because data maps can be defined across multiple content management systems, they are particularly effective at organizing varying metadata schemas into a common data model. This capability makes executing federated searches across multiple content management systems extremely simple to perform.

When a data map is associated with a search request, you can provide selection properties, selection criteria, and order by/sort specifications in terms of the data map element names instead of the repository-specific name; however, each name is valid. Content Integrator manages the necessary property substitutions as needed.

Using the example data map that is presented in Table 6-1 on page 299, you can use the element names firstname, lastname, and occupation in place of the actual repository property names F_NAM_STR, L_NAM_STR, and JOB_STR

respectively for Repository A and forename, surname, and employment_desc respectively for Repository B.

*Table 6-1 Example data map*

| Data map element | Property in Repository A | Property in Repository B |
|---|---|---|
| firstname | F_NAM_STR | forename |
| lastname | L_NAM_STR | surname |
| occupation | JOB_STR | employment_desc |

### 6.3.3 Session pools

Before interacting with content management systems, it is often necessary to perform a series of user initialization steps followed by logging on to the system, thereby setting up a user session connection. This series of operations is generally expensive in terms of processing requirements and tends to favor a stateful session model. In fact, most content management systems provide session-based APIs as the only means of operation. In an effort to minimize the costs involved with initializing and releasing these user sessions, Content Integrator provides a fully configurable session pooling option in the federated services architectural layer.

*Session pools* are a distributed user pooling mechanism that is made available for use by clients of Content Integrator. With session pools, client applications gain these abilities:

- Request user sessions connected to multiple content management systems
- Reuse user session connections in or among applications
- Limit the number of user session connections during heavy activity
- Minimize the number of user session connection initializations (logons and logoffs)
- Clean up residual user session connections due to unexpected client application termination

When configuring a session pool for use, system administrators might control these options, among other options:

- The maximum number of user sessions in the pool
- The maximum amount of time that a client application can keep a user session checked out of the pool
- The content management systems available to users created in the pool

- The maximum number of available user sessions maintained in the pool
- The minimum number of available user sessions maintained in the pool
- The timeout time for inactive user sessions

For client applications that serve a broad user community, such as publicly available Web sites or large-scale intranets, we strongly recommend session pools to lessen expensive or time-consuming user session initialization costs. For more information about session pools and the session pool server, see 6.5.4, “Session pool server” on page 312.

**Local session pools:** *Local session pools* are available to client applications that do not require the management of user sessions in a distributed environment. Local session pools perform exactly the same, otherwise.

## 6.3.4 Authentication and security

The general philosophy of Content Integrator with respect to authentication and security is to leave control to the underlying content management systems, ensuring that the investments made in each systems’ security model are preserved. Limitations arise, however, when operating with multiple systems in a federated environment. Each content management system involved in a federated request will likely require authentication credentials. To address this multiple logon concern, Content Integrator has provided a single sign-on (SSO) component.

Other security components made available in the Federation Services layer include several options for authorization security, as well as a logon encryption framework. We describe each of these topics in detail in the following sections.

### Authentication and the single sign-on system

You can use the Content Integrator SSO system to authenticate and manage users in enterprise environments where multiple content management systems are configured. User credentials, which are used for logging into various content management systems, can be stored, retrieved, and modified in a single, persistent location. Client applications can use the SSO system, and the credentials stored therein, to provide seamless access across disparate content management systems without requiring multiple user logons.

The SSO system includes an SPI that you can use if custom authentication models are required. As a convenience, a password vault reference implementation is provided as the default implementation of the SPI and is backed by an XML-based datastore. If needed, you can modify this datastore to use other persistence mechanisms, such as a relational database, using the SPI.

For more information, refer to 6.5.5, “Single sign-on server” on page 313.

### Authorization and the available authorization security models

As with authentication, Content Integrator does not supplant any existing authorization security infrastructure. After a user session has been authenticated by a content management system, the capabilities, or privileges, made available to that user session are managed by the authorization security model followed by the content management system.

In order to take advantage of the authorization security model that is made available by the various content management systems, Content Integrator associates one or more of the four security models with each connector implementation:

- The simple security model supports assigning only one principal to each of three privileges: read, write, and delete.
- The advanced security model supports access control lists (ACLs) with four privileges: read, update, delete, and change security.
- The enhanced security model exposes security features of the content management system, such as the security schema, hierarchical groups, ACLs, and system privilege types.
- The read security tokens model uses “read access tokens” to determine privileges for entities, such as users, groups, and roles.

**For more information:** Check the Content Integrator Information Center to determine which authorization security models are used by each connector implementation.

For clients developing custom connector implementations, any of these authorization security models are available to be used as required by the needs of the application, as well as the capabilities of the underlying content management system.

### Logon encryption framework

When distributed applications are developed using Content Integrator, client applications and associated connector implementations might be dispersed across untrusted networks. Therefore, authentication credentials are susceptible to interception and inappropriate use. To prevent these types of misuse, Content Integrator provides a logon encryption framework that you can incorporate into an application deployment.

The *logon encryption framework* is a simple-to-use framework consisting of an authentication bundle, a sealer for encrypting data, and an unsealer to decrypt sealed data. After user credentials have been provided to the authentication bundle in a client application, it is sealed by the sealer implementation before being forwarded along through the system. After a connector implementation receives a sealed authentication bundle, an unsealer is used to unseal and extract the authentication credentials as necessary.

The encryption framework permits a wide variety of encryption and decryption algorithms, such as public key cryptography and symmetric cryptography. You can implement sealers and unsealers, using the Content Integrator SPI, to meet any specification that is required by the client application.

**Default implementation:** A default encryption framework implementation is provided with Content Integrator using symmetric cryptography and the Blowfish encryption algorithm. This default implementation must be used *only* in situations where all participating computers in the system are trusted.

### 6.3.5 View services

The *view services* component is an optional module capable of converting content from its native format into formats that are displayable in standard Web browsers. Using view services, you can deploy content-based solutions capable of converting more than 150 document formats, such as Microsoft Word, Microsoft Excel, and Microsoft PowerPoint, into browser-ready HTML without requiring vendor-specific client software or viewers. Additionally, you can also convert image formats that are not natively handled in Web browsers, such as Tagged Image File Format (TIFF) and MO:DCA images, for easy viewing.

Included with view services is both a default standard image converter and a default standard content converter. As necessary, these converters transform native content formats into formats that can be viewed in a Web browser. The JPEG format is typically used for image files, and common office automation formats are typically converted to HTML. If preferred, however, you can configure other converters for use with view services using the Administration Tool. Also, Content Integrator even provides an SPI if custom converters are desired.

Beyond simple conversion, you can also configure view services to incorporate dynamic image processing capability, such as image rotation, image scaling, image enhancement, and more. You can also add conversion and streaming engines to support audio, video, and other content formats. All processing and conversions are keyed off the content's Multipurpose Internet Mail Extensions (MIME) type, which is configurable using the Administration Tool.

## 6.4 Developer and User Services layer

All functionality that is provided by Content Integrator is made available to software integrators and application developers using the Developer and User Services layer. The most visible of the three architectural services layers, the *Developer and User Services layer*, is home to the client development APIs and graphical user interfaces that enable the integration and configuration of Content Integrator into new or existing applications.

Solutions developers have full access to the range of capabilities exposed by content management systems using the Java-based Content Integrator API. For installations that are built around Web services architectures, the new service-oriented architecture (SOA) Web services API is available. Designed to be easily consumable, you can use either API to quickly integrate bidirectional access to enterprise content management systems.

The Developer and User Services layer also includes two graphical components: the Administration Tool and the common viewer. You use the Administration Tool to configure various aspects of an Content Integrator installation. For users who need a simple applet to view and manipulate native content, you can use the common viewer applet.

We describe in detail all four components comprising the Developer and User Services layer in the following sections.

### 6.4.1 Content Integrator API

The driving goal behind the design and development of Content Integrator has always been to provide a solution to the obstacles that are faced by businesses after a series of mergers or acquisitions. Often, it is discovered that customer and operational data is dispersed across various content management systems, and no single point of access is available. Therefore, investments in this decentralized data are not recouped, and business opportunities were lost. Content Integrator was developed to give enterprise solution developers the tools to overcome these hurdles; a single interface into these disparate systems is the mechanism to develop this single point of access.

At the heart of the Developer and User Services layer lies the Content Integrator Application Programming Interface (API). This Java-based API provides a full range of enterprise content management capability to solution developers creating new applications or enhancing existing systems that are already deployed. This API is intended to be used in a datastore-agnostic manner. After an enterprise application has been developed using Content Integrator,

introducing a new content management system into the environment simply involves adding a new connector.

The Content Integrator API is truly a collection of API sets that target key functional areas. The *Integrate API* is the most comprehensive of the APIs and the most often used. Enterprise content management system functionality made available through the Integrate API includes but is not limited to these functions:

- Logon/logoff
- Retrieve options
- Create options
- Update options
- Delete options
- Query and browsing options
- Security and records management options
- Check-in/check-out

For situations that require the dynamic, runtime configuration of an already-running Content Integrator deployment, you can use the *Configuration API,* which is a subset of the Java API. The Configuration API exposes a subset of the functionality that is available using the Administration Tool:

- Connector configuration options
- Data map options
- Session pool options

The second portion of the Content Integrator programming interface is the *Connector Service Provider Interface (SPI)* that was presented in 6.2.1, “Connectors and the Connector SPI” on page 293. The Connector SPI is used internally by Content Integrator to establish a consistent interface between the client API and the content management systems that are supported. Most application developers do not need to use this interface unless additional functionality is required that is beyond what has already been provided with a connector implementation, or a new connector development effort is underway.

### 6.4.2 SOA Web Services

A recent addition to the Developer and User Services layer is the SOA Web Services interface. SOA Web Services provides two significant benefits:

- For enterprise deployments that are already committed to a Web services-based architecture, you can use SOA Web Services to easily integrate the capabilities that are provided by Content Integrator.
- The SOA Web Services interface is easily consumable by a host of development languages. Application developers are not required to use Java.

Following industry standards for Web services development, the services that are provided by SOA Web Services are loosely coupled–each service functions completely and independently of any other service executed. These Web services are available for use:

- Retrieve options
- Create options
- Update options
- Delete options
- Query options

**Previous interface:** Prior versions of Content Integrator offered an interface based on the SOAP framework that was referred to as the Web Services API. This interface, *now deprecated*, more closely matches the fine-grained nature of the Content Integrator API.

### 6.4.3 Administration Tool

You configure an Content Integrator deployment using a stand-alone, Java-based application that is referred to as the Administration Tool. Presented in Figure 6-4 on page 306, you use the *Administration Tool* to configure and maintain the access services component, the view services framework, all connectors, data maps, session pools, and logging.

Archived

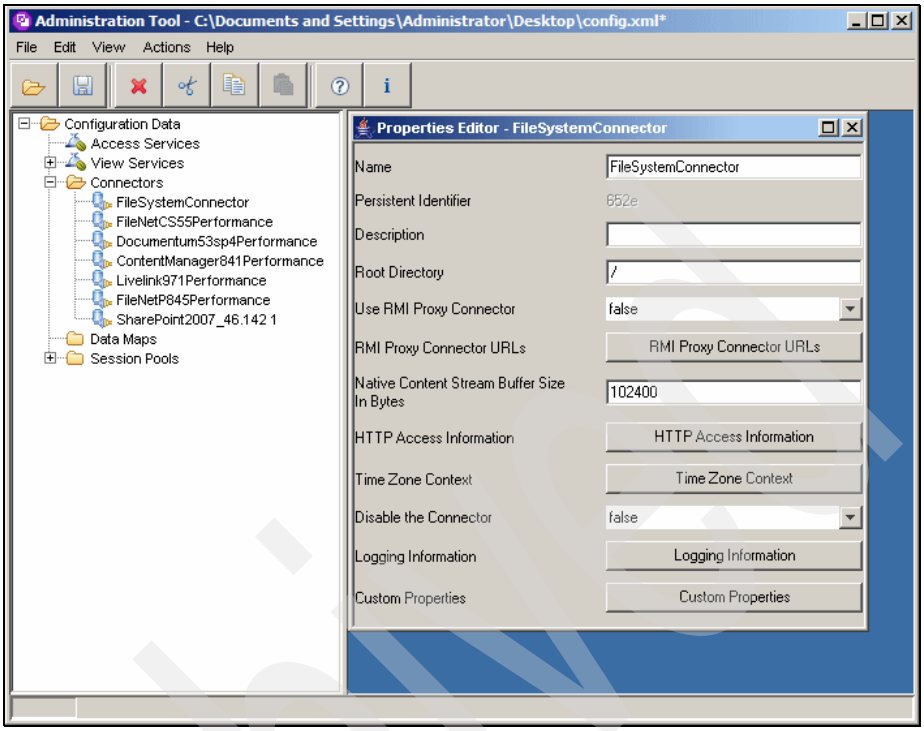



*Figure 6-4 Content Integrator Administration Tool*

There are two modes of operation available for use in the Administration Tool, local mode or connected mode:

- Using local mode, the Administration Tool interacts directly with the underlying XML-based configuration file (`config.xml`) that is stored on a local or shared file system. The local mode is particularly useful during the initial configuration efforts in a new deployment environment.
- Using connected mode, the Administration Tool finds and interacts with a running IBM Content Integration Configuration server. The connected mode is used in environments where client applications are already running and dynamic configuration updates are desired across the system. See 6.5.1, “Configuration server” on page 309 for more information.

The Administration Tool is accessible for users with disabilities. You can perform all of the tasks using keyboard shortcuts.

### 6.4.4 Common viewer

In many enterprise deployments, clients of Content Integrator often need to work with a wide variety of office automation and image formats. Instead of requiring that client applications incorporate a variety of native content management system browsers or viewers, you can use the common viewer applet.

Use the *common viewer* to view and manipulate various content formats without requiring additional content management system browsers, viewers, or libraries. You can zoom, rotate, and perform other image enhancements. And depending on the capabilities of the underlying content management system, document annotations might be supported.

The common viewer recognizes the following MIME types:

- AFP (not large object)
- BMP
- CALS
- DCX
- DICOM
- GIF
- IOCA
- JFIF
- JPEG
- JPG
- LINE (not large object, treated as ASCII text)
- MO:DCA
- PCX
- PDF
- PNG
- PTOCA
- TIF
- X-CALS
- X-DCX
- X-PCX
- X-TIFF

You can use the common viewer applet in one of several ways: you can deploy it as a stand-alone applet, deployed as an applet embedded in an existing Web application, or run as a simple Java Swing application running outside of a Web server.

The common viewer is accessible for users with disabilities. You can perform all of the tasks using keyboard shortcuts.

Archived

Figure 6-5 presents the common viewer running as a simple Java Swing application.

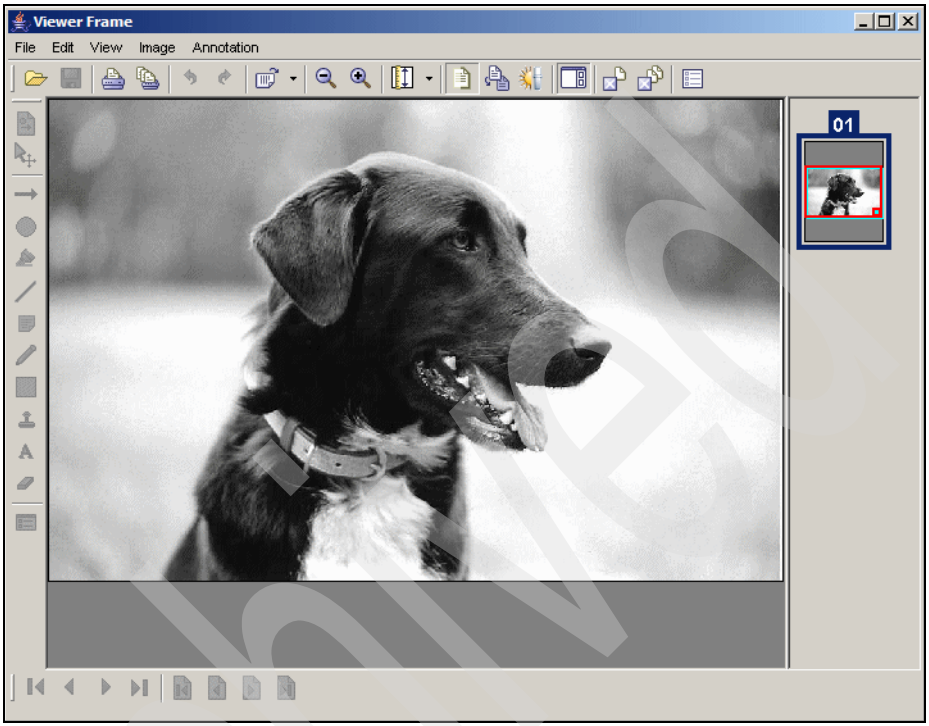



*Figure 6-5 The common viewer running as a simple Java application*

## 6.5 Additional services

Content Integrator provides additional modules, or *services*, that are either server extensions to components already discussed or are additional services that are not easily characterized by the three architectural layers presented earlier in this chapter. The additional modules include these services:

- Configuration server
- Logging server
- RMI proxy connector server
- Session pool server
- Single sign-on (SSO) server

We discuss each of these services in detail in the following sections.

### 6.5.1 Configuration server

Content Integrator provides a graphical utility that is called the Administration Tool, which is used to configure the key components that make up an Content Integrator deployment. This configuration information is persisted in an XML file on the underlying file system (`config.xml`). Direct access to the configuration information contained in this file is generally not an issue for small deployments or in testing environments where only a few clients are running–you can deposit the file in a shared folder that is accessible by all clients. For larger deployments, however, this method of configuration sharing might not be a viable option.

A configuration server has been provided with Content Integrator to handle the hosting and dispersal of configuration information. Upon startup, the configuration server consumes the configuration details from the XML file and makes the details available to clients requesting them. Additionally, dynamic updates are possible, because changes that are made to configuration information can be made “in memory” using the Administration Tool.

### 6.5.2 Logging server

The logging server is an optional service that is provided by Content Integrator. The logging server provides a file logging mechanism that is used to capture informational, warning, and error messages, depending on the specified configuration, as they occur in a running environment. You can initiate and configure any number of logging servers to output to one or more files. Use the Administration Tool to configure all of the logging specifications.

**Using a central logging server:** In a distributed environment, it might be beneficial to have dispersed connector deployments report error messages to a central logging server. However, be mindful of the additional network traffic that can be incurred.

### 6.5.3 RMI proxy connector server

For clients that are built on Content Integrator that require few computing resources in terms of memory usage or processing throughput, you can deploy connector implementations “in-process” within the same JVM that is hosting the client application. In many instances, however, it is beneficial to develop a deployment architecture whereby connectors are hosted in their own JVMs that are capable of being tuned to meet the needs of the enterprise system. To support this deployment option, you can use RMI proxy connector servers.

Archived

Developed using Java Remote Method Invocation (RMI) technology, RMI proxy connector servers simply host connector implementations, and their associated configurations, for an Content Integrator environment. When deployed in this manner, RMI proxy connectors can service multiple distributed, client applications. More importantly, you can tune the associated JVMs for any number of Java options, including but not limited to memory usage, garbage collection, process priority, and timeouts. The associated JVMs can even be deployed on dedicated computing resources for maximum throughput capabilities.

We present sample connector deployment options in the following series of figures. We present the simplest connector deployment option in Figure 6-6. In Figure 6-6, the connector instance is hosted within the same JVM as the simple client application. This deployment option is generally only used in testing situations or in environments where computing resources are not a limiting factor.

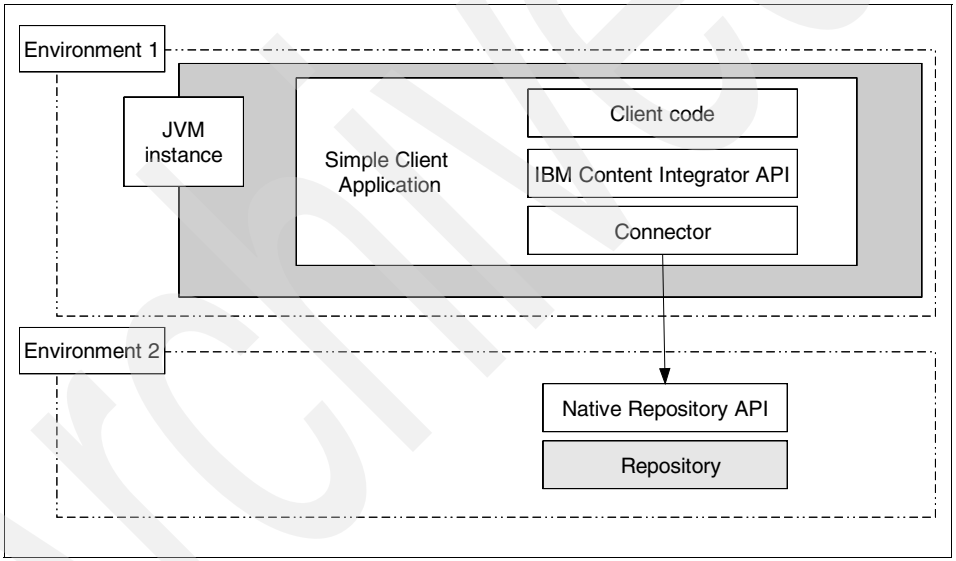



*Figure 6-6 Simple "in-process" connector deployment*

For even the smallest client installations, it is extremely common to deploy the connector instance in an RMI proxy connector server, as presented in Figure 6-7 on page 311. This scenario allows system administrators the ability to tune the JVM instances as needed. Note that in this example the RMI proxy connector server is still hosted in the same environment as the client application.

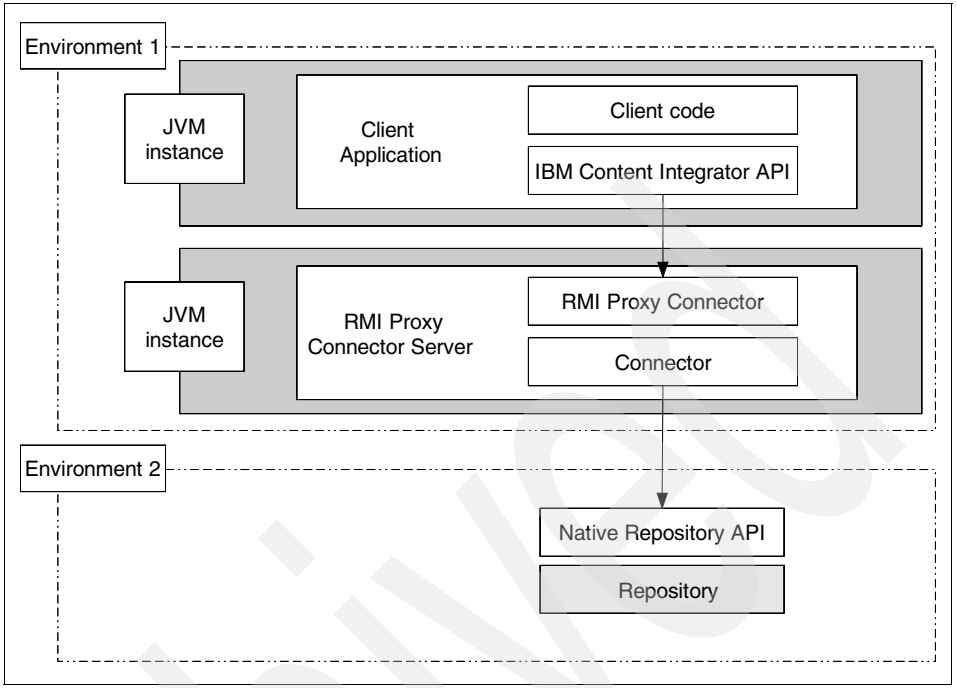



*Figure 6-7 Simple RMI proxy connector server deployment*

For enterprise-level applications that are developed with Content Integrator, we recommend to deploy RMI proxy connector servers on their own computing environment, as presented in Figure 6-8 on page 312. In this deployment scenario, not only can administrators tune individual JVMs for better efficiency and throughput, you can also administer entire computing environments to serve the needs and requirements of each enterprise application. Additionally, connectors that are deployed in this manner can serve multiple clients at no additional configuration or administration cost.

Archived

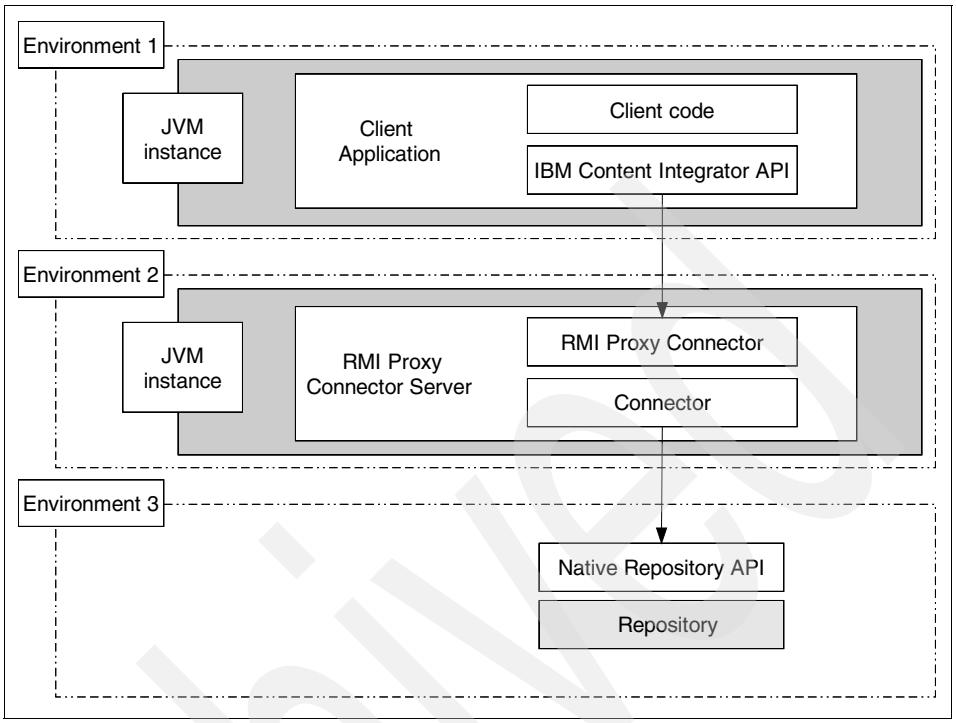



*Figure 6-8 Recommended enterprise-level RMI proxy connector server deployment*

## 6.5.4 Session pool server

Developed using Java Remote Method Invocation (RMI) technology, session pool servers are used to provide support for *distributed* session pools. When session pool servers are configured using the Content Integrator Administration Tool, you can host a session pool by one or more session pool servers where user session connections are managed and controlled across any number of client applications.

Figure 6-9 on page 313 presents a simple example. Two client applications use the same session pool server for user sessions. Using the Developer and User Services layer, each client application requests a user from the session pool. Respecting any configured properties for the session pool, the session pool server handles each client request, either creating a new user session or reusing an already established, though currently inactive, user session. When the client application no longer needs the user session, the user is released back into the session pool.

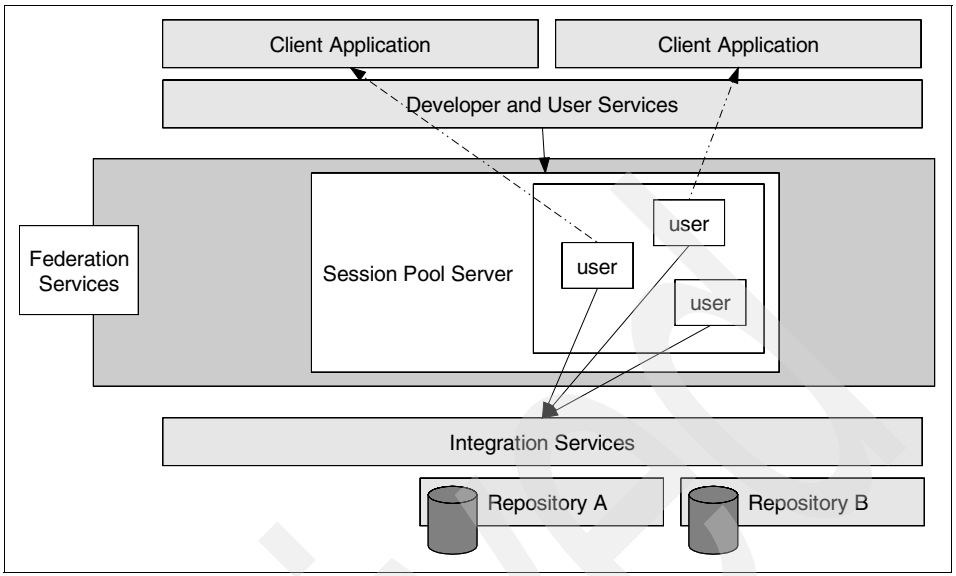



*Figure 6-9 Simple session pool server example*

For more information about session pools, see 6.3.3, “Session pools” on page 299. For an advanced discussion about session pools, see 8.3.1, “Session pools” on page 412.

### 6.5.5 Single sign-on server

A single sign-on (SSO) server is provided by Content Integrator for use when client applications require remote, read-only access to an authentication system implementation. Complex configuration requirements might be necessary in authentication system implementations to access cryptography libraries and keys, as well as access to remote authentication systems. In these scenarios, it might be ideal to configure an SSO server that can be used by many client applications instead of requiring that all of the clients be configured. Client applications can then access the SSO system that is hosted by the server using a simple client service.

For an advanced discussion about SSO, see 8.3.2, “Single sign-on” on page 421.

## 6.6 General design principles

Throughout the design and development of Content Integrator, a collection of design principles has been adopted to insure data integrity, whether managed through Content Integrator or directly with the content management systems in use. Additionally, investments in the native toolsets that are provided by content management systems are not compromised in any way–users of the native tools do not detect Content Integrator's presence. To meet these requirements, the following design principles have been adopted by the product:

- Users of Content Integrator are only exposed to content management system abstractions.

  In general, users working with content are not required to have any knowledge of the content's physical location. Likewise, application developers must think of content management system concepts abstractly, because there are no content management system-specific APIs exposed through Content Integrator. Working in these terms limits exposure to the vagaries of content management system specifics. Users only have to learn one set of behaviors. Developers only need one set of development APIs.

- Content management system functionality is never supplemented or enhanced.

  There are hundreds of content management systems in use today, with a wide variety of functional capability (and limitations). Content Integrator never attempts to compensate for these limitations. Common functionality is exposed through the API. Use the repository profile to determine the available functionality.

- Content management systems manage their own authentication and authorization.

  Content Integrator never imposes any of its own authentication, authorization, or security. All security is managed by the underlying content management systems.

- Neither content nor any of its associated metadata is ever stored in Content Integrator.

  Content Integrator was never designed to be a content management system. Any content, or any metadata that is associated with that content, must be accessible using the native tools that are provided with the underlying content management systems. No content or associated metadata is persisted in Content Integrator.

Archived

## 6.7 Recent product enhancements

So far in this chapter, we have described the architecture of Content Integrator Version 8.5.1. For clients who use an earlier version, we present a list of architectural changes and functional enhancements in versions 8.4 and 8.5. These changes were significant and have greatly improved the functionality and consumability of the product. The resulting benefits make upgrading an attractive proposition.

If you currently use a version earlier than versions 8.4 or 8.5, we recommend reviewing the enhancements in this section and upgrading to the latest version soon.

Each of the following enhancements is further explained in a subsequent section:

- Version 8.4:
  - Direct mode
  - Connector and API enhancements
  - Configuration API
  - Common Viewer
  - Enhanced security model
  - Connector plug-ins
- Version 8.5:
  - Full support for direct mode
  - SOA Web Services
  - Native content streaming
  - Connector enhancements
  - New SSO reference implementation
  - New Data Map Designer
  - Connector War file deployment option

**Note:** In Version 8.5, the product was renamed from *WebSphere Information Integrator Content Edition (IICE)* to *Content Integrator*. For consistency reasons, we use the new product name throughout this section.

There are other IBM products that have a similar name, but they have no relationship to Content Integrator:

- WebSphere Information Integrator (WebSphere II) allows federation and transformation of data from heterogeneous database systems.
- Information Integrator for Content allows federation of content from IBM content management systems, such as IBM Content Manager, IBM Content Manager OnDemand, or IBM Image Plus for OS/390®. Information Integrator

Archived

for Content has a similar conceptual approach as Content Integrator, but it is mainly used as a programming interface to IBM Content Manager.

## 6.7.1 Product enhancements in Version 8.4

Content Integrator Version 8.4 provides the following product enhancements.

### Direct mode

*Direct mode* is the name of a simplified deployment option that was introduced in Version 8.3 and enhanced in Version 8.4. Direct mode exposes important components, such as Access Services, View Services, and connectors as Plain Old Java Objects (POJO) without any Enterprise JavaBeans (EJB) dependency. If direct mode is enabled, you can run Content Integrator in a Java virtual machine, and you do not need to deploy it on an application server. In direct mode, Content Integrator is a simple set of Java libraries rather than an EJB application. Direct mode greatly reduces setup time and complexity.

The counterpart of direct mode is *server mode*. With server mode enabled, you have to deploy Content Integrator on an application server. As shown in Example 6-1, the following JVM property determines the operation mode of Content Integrator.

*Example 6-1 Configuring direct mode or server mode*

```
// enable direct mode
vbr.as.operationMode=direct

// enable server mode
vbr.as.operationMode=server
```

Direct mode not only simplifies the installation and deployment of Content Integrator, but it also simplifies the configuration of client applications. In direct mode, a client simply adds Content Integrator jars to its classpath and sets two JVM properties. In contrast, in server mode, a client also needs to add the client libraries of the application server to its classpath and to configure the location of the application server through Java Naming and Directory Interface (JNDI) properties.

In direct mode, it is still possible to embed Content Integrator in a Web application or enterprise application and deploy the composite application on an application server.

No matter if you choose direct mode or server mode, there are two choices of where to host the connector: You can either host a connector in a separate JVM

in an *RMI proxy connector server*, or you can run the connector locally in the same JVM as AccessServices[1] [2]. This choice is determined by the flag "Use RMI proxy connector" in the connector configuration in the Administration Tool. Note that Version 8.5 introduces a third connector deployment option.

## Connector and API enhancements

Version 8.4 contains many functional connector and API enhancements. For example, new search operators were introduced, and support for minimum, maximum, and default property values was added. Also, query capability was enhanced to allow for multiple search containers and sorting by multiple properties. Finally, the implementation of item retrieval and query execution was optimized in important connectors. The following page in the information center contains a complete list of connector and API enhancements:

http://publib.boulder.ibm.com/infocenter/ce/v8r4/topic/com.ibm.discovery.ci.product.doc/iiyva_new_84.htm

## Configuration API

The Configuration API is a Java programming interface that allows client applications to programmatically modify the configuration of Content Integrator. For example, a client application can use the Configuration API to create a new connector, configure it, and connect to a repository. Previously, the configuration of a new connector was only done manually in the Administration Tool. The introduction of the Configuration API is a major consumability milestone. The combination of direct mode and the Configuration API makes it extremely convenient for other applications to embed Content Integrator and gain instant access to remote repositories. See 8.3.3, "Configuration API" on page 426 for a detailed discussion about the Configuration API.

## Common Viewer

The Common Viewer is a Java Applet that can display, manipulate, and print images, PDFs, and other types of native content. It also supports various operations on annotations. The Common Viewer replaces the Viewer Applet of prior versions. You can run it as a stand-alone Java application or embedded in a Web page. You can obtain more details in 6.4.4, "Common viewer" on page 307.

---

[1] In server mode, there are certain connectors that cannot run in the JVM of an application server. These connectors use native components to communicate with the repositories, which is not allowed by the Java 2 Platform, Enterprise Edition (J2EE) specification. These connectors are FileNet CS/IS, Microsoft NT File System, Lotus Notes, DominoDoc, Hummingbird, and OnDemand Web Enablement Kit. This limitation does not exist in direct mode.

[2] Theoretically, server mode offers an additional option for connector deployment: a heavily used connector can run as a separate EJB application in a server cluster by itself.

### Enhanced security model

The Enhanced security model was introduced to allow fine-grained access to repository security settings. Clients can use the Enhanced security model to retrieve and modify permissions on individual repository items through access control lists (ACLs) and native repository privileges. The model also exposes users, groups, group memberships, and document owners. It is the most elaborate security model of the Content Integrator Java API. To learn more about the Enhanced security model, see 8.3.5, “Security models” on page 430.

### Connector plug-ins

Connector plug-ins provide a formal mechanism for extending the functionality of Content Integrator. For more information, see 9.3, “Connector plug-ins” on page 488.

### Removal of components

The Web client and Web components were removed from the product. The Web client is available as a sample at the IBM developerWorks® Web portal:

http://www.ibm.com/developerworks/data/library/techarticle/dm-0906iciwebclient

## 6.7.2 Product enhancements in Version 8.5

Content Integrator Version 8.5 provides the following product enhancements.

### Full support for direct mode

Version 8.5 extends support for direct mode to all of the components of Content Integrator. EJB dependencies are completely removed, and the product no longer requires deployment on an application server. Server mode is removed, and direct mode is the only available operation mode. Hence, the client JVM property vbr.as.operationMode is no longer required.

To achieve scalability in an environment with a high number of parallel client requests and failover capability, we recommend the RMI proxy connector server deployment option. It allows connectors to be distributed on multiple JVMs and machines.

For more details about deployment recommendations, see Appendix A, “Planning and sizing the Content Integrator system” on page 513.

### SOA Web Services

The SOA Web Services are a new set of Web services introduced in Version 8.5. In contrast to the existing Web Services API, each SOA Web service is stateless

and supports a range of capabilities related to one use case. For example, the RetrieveDocument Web service supports the retrieval of an item's metadata, native content, and security information in a single request. See 6.4.2, "SOA Web Services" on page 304 for a list of available SOA Web Services.

### Native content streaming

*Native content streaming* is a new mechanism for the transfer of native content between the connector and the client application. The new mechanism exposes the native content bytes of an item as an InputStream to the client. A request for a large file is executed as multiple requests for smaller chunks, thus creating a streaming effect. Native content streaming allows for the transfer of large files and prevents out of memory errors in the connector JVM and the client JVM. See 8.2.4, "Retrieval of metadata and native content" on page 401 for a code sample using the new streaming methods.

### Connector enhancements

Multiple connectors were enhanced to support records management through IBM FileNet Content Services. Enhancements include the support for locking down and unlocking a content item and for deleting a particular content version. Additionally, support for the Enhanced security model was added to the IBM FileNet P8 connector.

### New Single Sign-On reference implementation

A new XML-based Single Sign-On reference implementation was added, and the existing reference implementation and its underlying POET data store (a proprietary object-based database system, currently called FastObjects t7) were removed. Additionally, to support credential stores other than the default XML store, a Persistence SPI was introduced. The Persistence SPI allows you to plug in an adapter to a credential store of your choice while reusing those parts of the reference implementation that are independent of the persistence mechanism.

### New Data Map Designer in the Administration Tool

The Data Map Designer in the Administration Tool was replaced with a new user friendly interface that simplifies the creation and editing of data maps. The new designer also supports changing the item class that is associated with a connector in an existing data map.

### Connector War file deployment option

A new deployment option was introduced that allows select connectors to be hosted in a servlet container of a Web server. In this deployment scenario, an administrator can start and stop the connector remotely through the Web server Web interface in case no direct machine access is available.

### Removal of components

The following components were removed from the product: Subscription Event Services, Virtual Repository API, POET Data Store, Relate Data Store Services, and the Services Monitor Web application. As a consequence, the installation footprint is reduced, and the product configuration is simplified.

Archived

7

# Content Integrator installation and configuration

This chapter describes the installation and configuration of IBM Content Integrator. In addition, we provide detailed connector configuration instructions for the most frequently used content management systems.

We cover the following topics in this chapter:

- Installing Content Integrator
- Configuring Content Integrator
- Configuring Content Integrator connectors:
  - EMC Documentum connector configuration
  - Hummingbird Document Management connector configuration
  - IBM Content Manager connector configuration
  - IBM FileNet Content Services connector configuration
  - IBM FileNet Content Manager connector configuration
  - Microsoft Windows SharePoint connector configuration
  - Open Text Livelink connector configuration

© Copyright IBM Corp. 2010. All rights reserved.

## 7.1 Installing Content Integrator

Beginning with Content Integrator Version 8.5, a Java/Java 2 Platform, Enterprise Edition (J2EE) application server is no longer needed to support Content Integrator usage. Removing the complexities of application server deployment now provides us with an extremely simple installation procedure.

We present the typical installation for Content Integrator Version 8.5 on a Microsoft Windows platform. (Installation procedures for other operating systems follow the same general flow.) Follow these steps:

1. Depending on how you received the Content Integrator Version 8.5 distribution, you might need to extract the installation files. If you have a single executable file, such as `C1RB5ML.exe`, extract the installation files by double-clicking the file and providing a temporary holding location.
2. To start the Content Integrator Version 8.5 installation, execute the file `setupWin32.exe` by double-clicking it.
3. When requested, choose a language to use for the installation wizard. English is the default language.
4. After the Content Integrator Version 8.5 splash window, the welcome panel appears, as presented in Figure 7-1. Click **Next**.

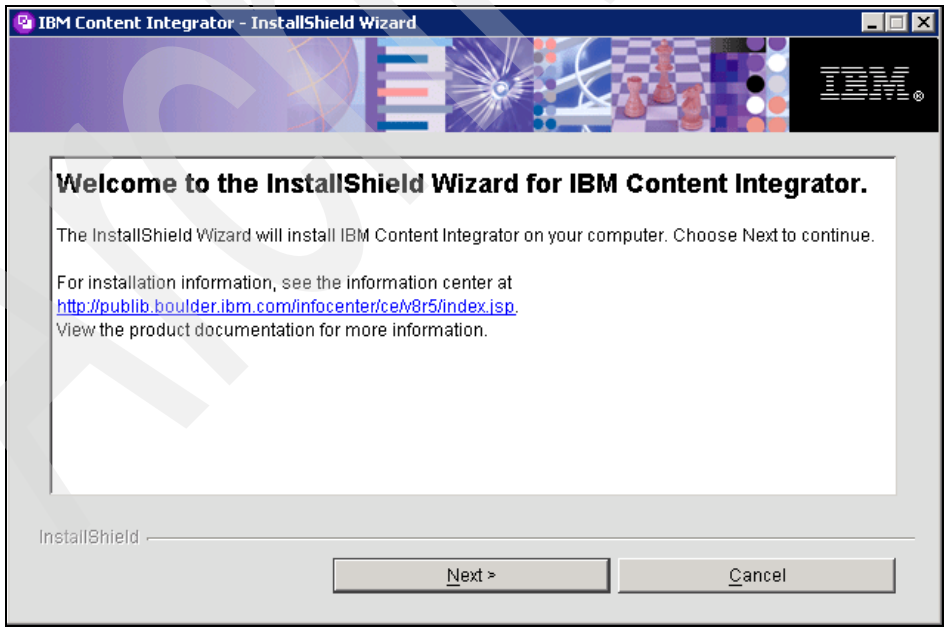



*Figure 7-1 Content Integrator welcome panel*

5. When the license panel appears, as presented in Figure 7-2, select the option “I accept the terms in the license agreement” and click **Next**.

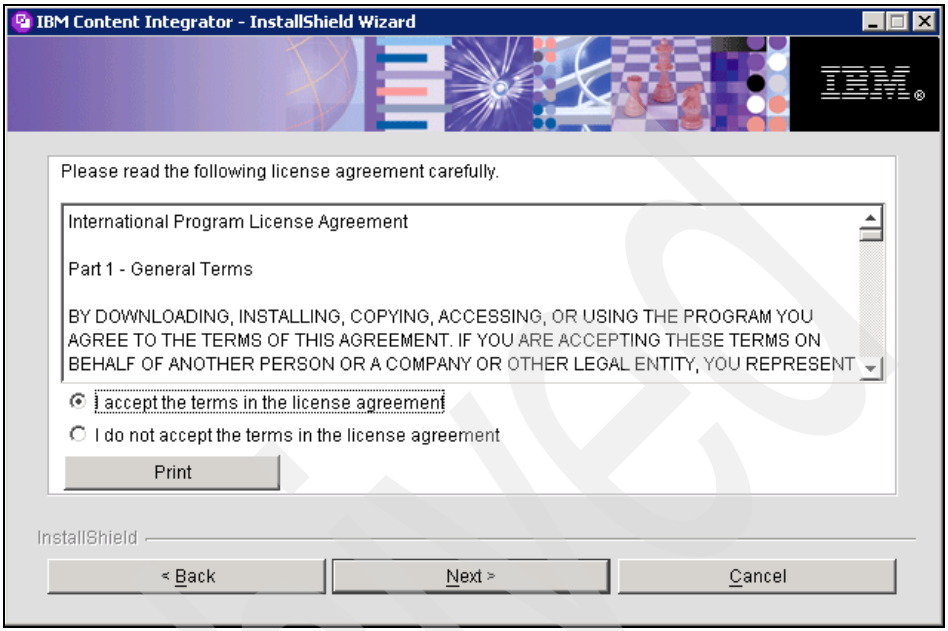



*Figure 7-2 Content Integrator license panel*

6. When the install location panel appears, as presented in Figure 7-3 on page 324, enter the desired install location, or accept the default location. Then, click **Next**.

Archived

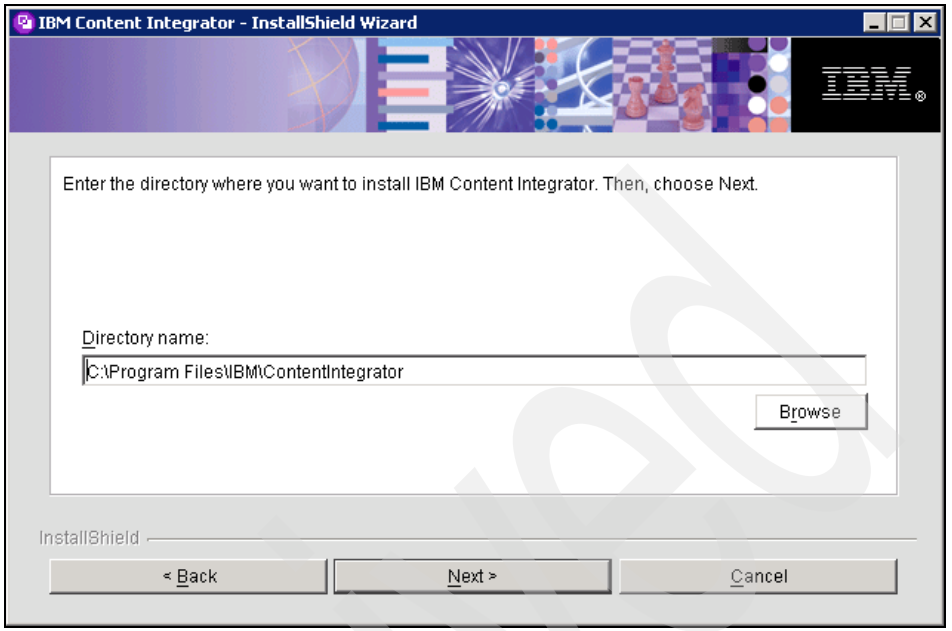



*Figure 7-3 Content Integrator install location*

7. Next, you are asked to choose an installation type, as presented in Figure 7-4 on page 325. Requesting a “Full” installation gives you all of the Content Integrator components. “Connectors Only” installations are generally used for systems that run Remote Method Invocation (RMI) Proxy Connectors only. Since Version 8.5, the differences between these options are minimal.

   Select the **Full** installation type, and then, click **Next**.

Archived

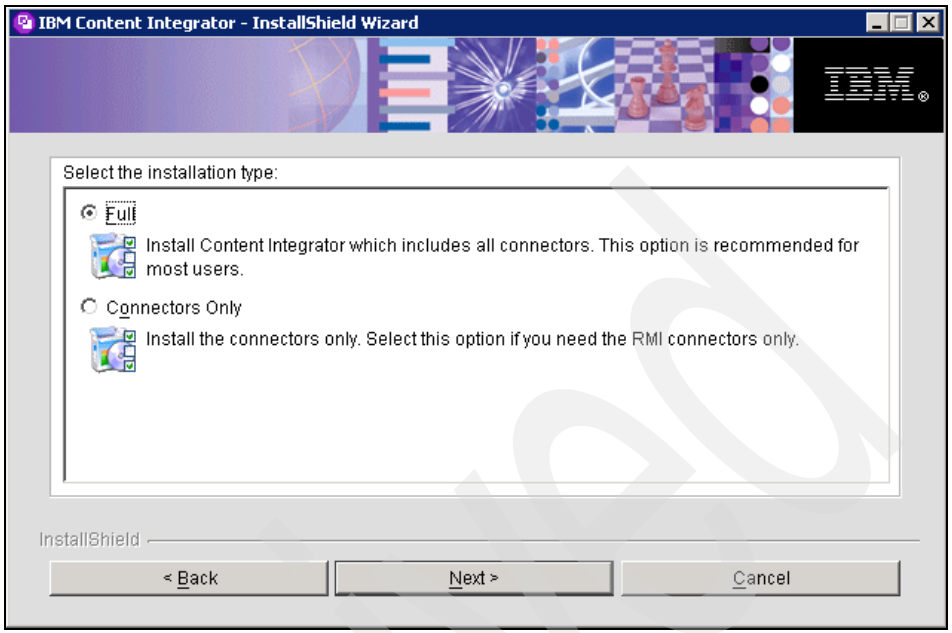



*Figure 7-4 Content Integrator installation type*

8. An installation summary panel, as shown in Figure 7-5 on page 326, appears. Review the summary, and click **Install**.

Archived

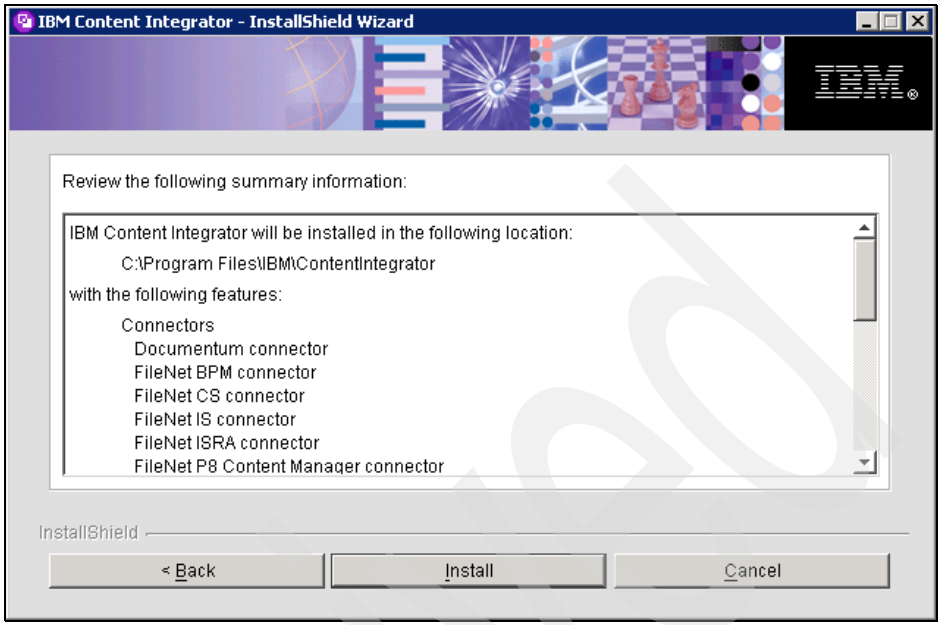



*Figure 7-5 Content Integrator installation summary*

9. Soon, the installation complete panel, as shown in Figure 7-6 on page 327, appears. Click **Finish**.

Archived

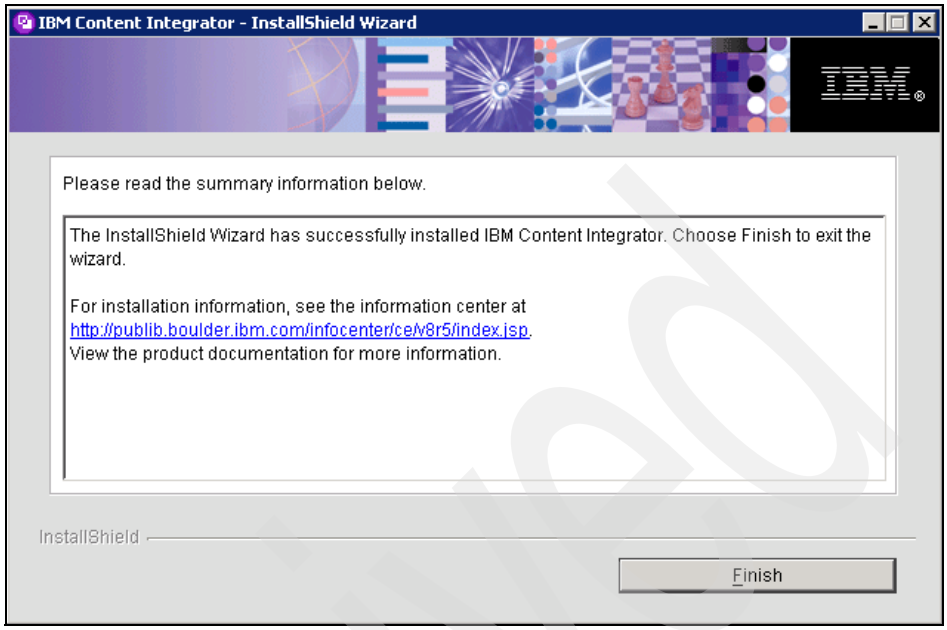



*Figure 7-6 Content Integrator installation complete*

10. After installation and before configuring your deployment using the Administration Tool, you need to modify the *JAVA_HOME* variable that is used in the startup scripts to direct them to a valid Java 2 installation. This variable assignment is made in the `config.bat` or `config.sh` file that is located in the `/bin` directory of your Content Integrator install location.

    For example, if you have installed Java in the following installation directory, `C:/java/ibm-java2-sdk-50`, make the following modification to your script file:

    ```
    SET JAVA_HOME=C:/java/ibm-java2-sdk-50
    ```

Your Content Integrator Version 8.5 installation is now complete.

## 7.2 Configuring Content Integrator

You perform the configuration of an Content Integrator deployment using a stand-alone, Java-based application that is referred to as the Administration Tool. Presented in Figure 7-7 on page 328, you use the *Administration Tool* to configure and maintain the access services component, the view services framework, all connectors, all data maps, all session pools, and logging.

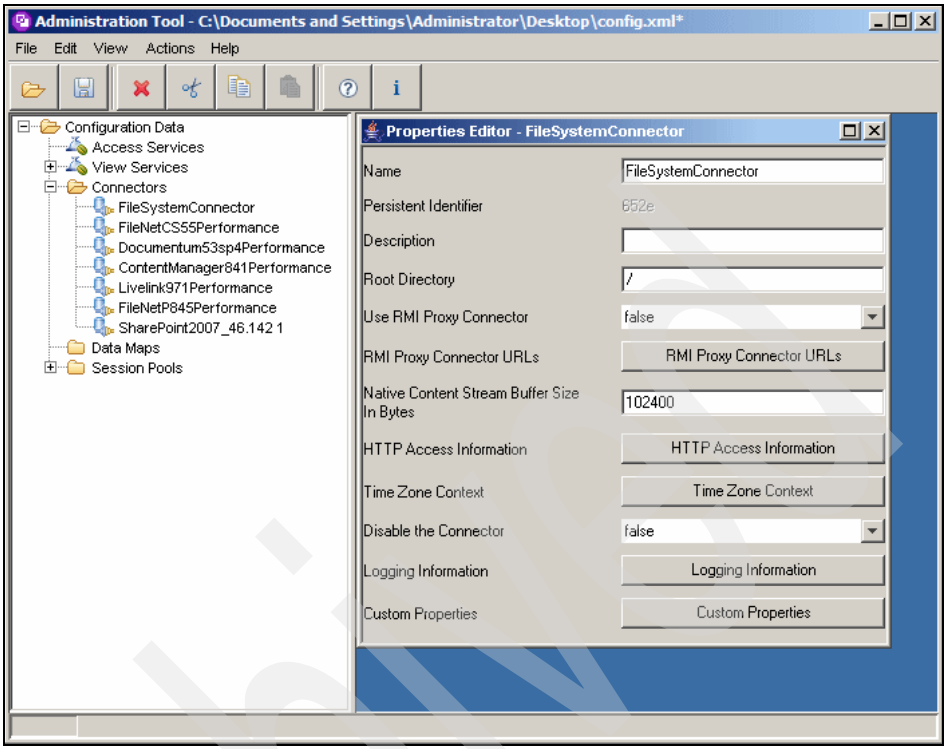



*Figure 7-7 Content Integrator Administration Tool*

There are two modes of operation that are available for use in the Administration Tool, local mode or connected mode:

- Using local mode, the Administration Tool interacts directly with the underlying XML-based configuration file (`config.xml`) that is stored on a local or shared file system. The local mode is particularly useful during the initial configuration efforts in a new deployment environment.
- Using connected mode, the Administration Tool finds and interacts with a running IBM Content Integration Configuration server. You use the connected mode in environments where client applications already run and dynamic configuration updates are desired across the system. See 6.5.1, “Configuration server” on page 309 for more information.

The Administration Tool is accessible for users with disabilities. You can perform all of the tasks using keyboard shortcuts.

For details regarding specific component configuration, see the Content Integrator Information Center. Instructions that describe in detail the configuration of the most frequently used Content Integrator connectors follow this section.

## 7.3 Configuring Content Integrator connectors

After an Content Integrator installation, the most frequently configured components are connectors. *Connectors* allow client applications written against the Content Integrator API to communicate to established content management systems. Depending on the particulars of the content management system, specifically the consumability of the native repository's API, connector configurations range in complexity from extremely simple to moderately complex. Another consideration is whether you want an RMI proxy connector.

In the remainder of this chapter, we present a brief discussion of the RMI proxy connector setup before presenting general connector configuration details covering the properties that apply to all connectors. Finally, we present detailed connector configuration instructions covering our most frequently used content management systems:

- EMC Documentum
- Hummingbird Document Management
- IBM Content Manager 8
- IBM FileNet Content Services
- IBM FileNet Content Manager
- Microsoft SharePoint
- Open Text Livelink

### RMI proxy connectors

In most enterprise deployments of Content Integrator, a recommended practice is to utilize RMI proxy connectors to allow the most flexibility with deployment architecture, resource management, and performance tuning. For an architectural overview of RMI proxy connectors and RMI proxy connector servers, see 6.2.3, "Remote Method Invocation (RMI) Proxy Connector" on page 296 and 6.5.3, "RMI proxy connector server" on page 309.

Archived

Typically, there are only two additional steps that are necessary *after* the regular setup and configuration of your connector to use an RMI proxy connector:

1. Modify and start the appropriate RMIBridge script file to initiate the RMI proxy connector server:

   a. Edit the appropriate file, either the `/bin/RMIBridge.bat` or `/bin/RMIBridge.sh` file, that is located in the Content Integrator installation directory.

   b. Insure that the content management system API libraries are available to the RMI proxy connector server environment. These libraries are identified in each connector configuration section that follows in this chapter or in the Content Integrator Information Center:

      - For connectors communicating to systems using Java libraries, insure that the necessary Java libraries can be found in the RMI proxy connector server classpath. Generally, this step is covered if the JAR files have been added to the *ICI_HOME*`/lib` directory. A helper script file, `config.bat` or `config.sh`, picks these library files up and adds them to the classpath.

      - For connectors communicating to systems using other API technologies, such as COM, the native libraries are made available to the environment *PATH* variable in the `config.bat` or `config.sh` file. In a Windows environment, modify the *PATH* variable as necessary to include the native libraries that your RMI proxy connector server needs. (Equivalent variables in other environments: Linux/UNIX, use *LD_LIBRARY_PATH*, and in AIX®, use *LIBPATH*)

   c. If necessary for your environment, update the default RMI registry port number to use for this RMI Connector Proxy server instance. The default value is 1251:

      ```
      set VBR_RMIPORT=1251
      ```

   d. Optional: It helps to add a title that is used by the script to help identify the connector and the port number associated with the script, for example:

      ```
      title FileNet RMI Proxy Connector port 1251
      ```

2. Provide the necessary RMI proxy connector configuration options in the connector configuration:

   a. When configuring a connector for use within Content Integrator, one of the available common connector configuration properties is the Use RMI proxy connector property. Set this property to true to indicate that the connector will be hosted in an RMI proxy connector server.

   b. Additionally, a valid RMI proxy connector URL is required. Select **RMI Proxy Connector URLs** to add and modify these URLs. Look to the

“Common connector properties” on page 331 for more information about how to set these properties.

**RMI proxy connector servers and firewalls:** In certain environments, it might be necessary to host an RMI proxy connector server behind a firewall. The nature of RMI is to use randomly assigned port values for RMI communication. You can request that the RMI proxy connector server use a specific RMI port by using the *vbr.rmi.port* environment variable and assigning it to a port value that has been opened through the firewall:

```
-Dvbr.rmi.port=1255
```

## Common connector properties

The following connector properties, which are presented in Figure 7-8, are available to all Content Integrator connectors.

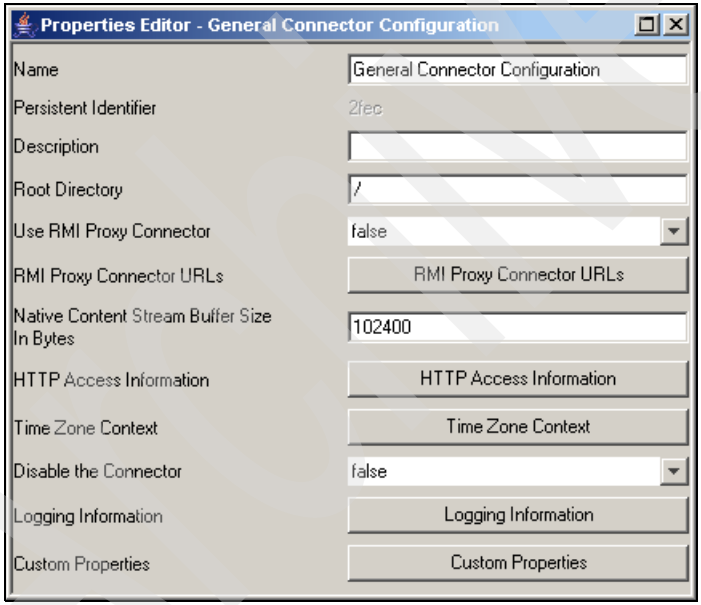



*Figure 7-8 Properties that are common to all Content Integrator connectors*

We list the properties and their definitions:

- **Name**

  The name of the connector configuration. This name is used to identify a specific connector configuration within the Content Integrator implementation and is often used as a parameter to several of the Content Integrator API and Service Provider Interface (SPI) operations.

- **Persistent Identifier**

  This identifier is a unique identifier that is associated with the connector configuration. The persistent identifier property is used internally. This read-only property is assigned when a new connector configuration is requested.

- **Description**

  This property is the description of the connector. This property is not used by Content Integrator for processing.

- **Use RMI Proxy Connector**

  The Use RMI proxy connector property enables or disables the use of an RMI proxy connector for the connector configuration. If set to true, provide at least one enabled RMI proxy connector URL for the RMI proxy connector URLs property. The default value is false.

- **RMI Proxy Connector URLs**

  This property is a list of one or more URLs to RMI proxy connector servers. The URLs are created or modified using the RMI proxy connector URLs editor. Initiate the editor, as presented in Figure 7-9, by selecting **RMI Proxy Connector URLs**.

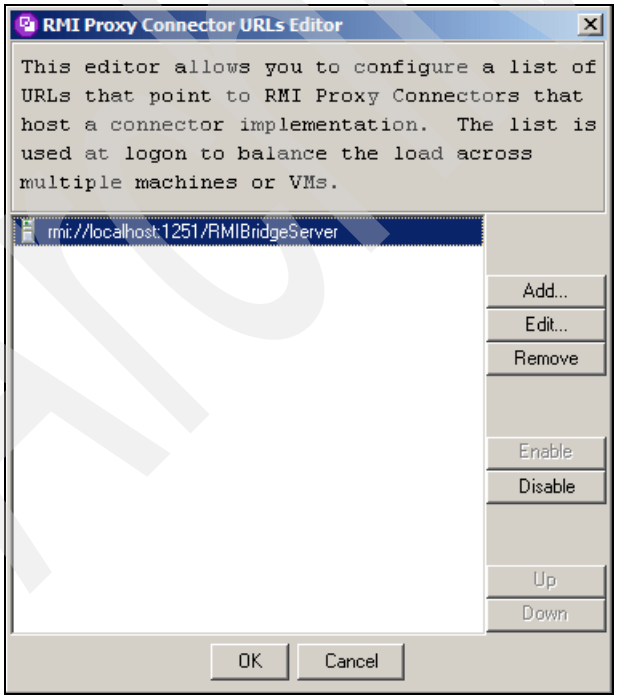



*Figure 7-9 RMI Proxy Connector URLs Editor window*

The format for the URL is `rmi://machine:port/RMIBridgeServer` where *machine* is the server name or IP address and *port* is the RMI registry port value to use. Examples:

```
rmi://9.18.33.132:1251/RMIBridgeServer
rmi://localhost:1253/RMIBridgeServer
rmi://eng-machine:1251/RMIBridgeServer
```

This property must provide at least one enabled RMI connector proxy service URL if the value of the Use RMI proxy connector property is set to true.

- **Native Content Stream Buffer Size In Bytes**

  This property is the buffer size to use for content streaming in Content Integrator. This value is in bytes. The default value is 102400.

- **HTTP Access Information**

  HTTP Access provides an alternative mechanism for native content retrieval and upload. To configure HTTP access server information for the connector, select **HTTP Access Information** to use the HTTP Access Information Editor, as presented in Figure 7-10. See the Content Integrator Information Center for more information.

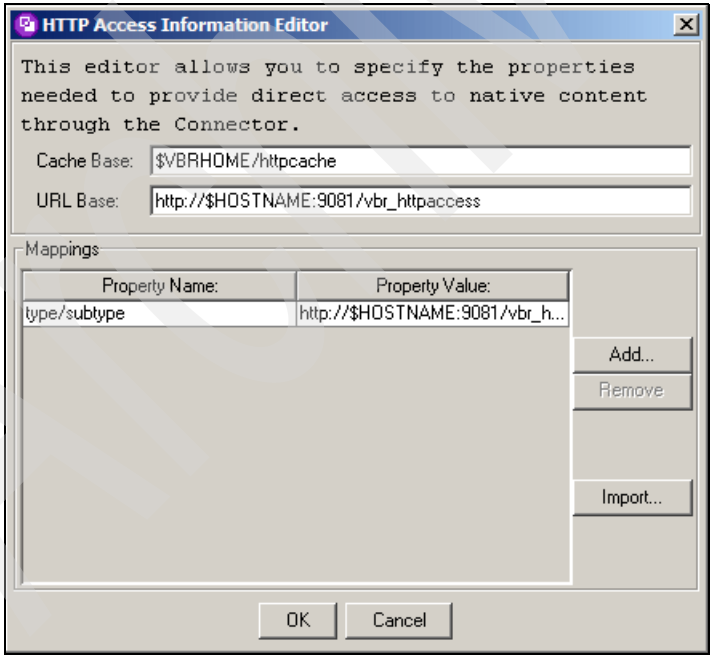



*Figure 7-10 HTTP Access Information Editor window*

- **Time Zone Context**

  By default, an Content Integrator connector assumes that it is hosted in a JVM that is running in the same time zone as the content management system with which it communicates. In certain enterprise deployments, this assumption might not be correct. A time zone context can be associated with the connector configuration using this property.

  To set the time zone context for the connector configuration, select **Time Zone Context** to use the Time Zone editor, as presented in Figure 7-11.

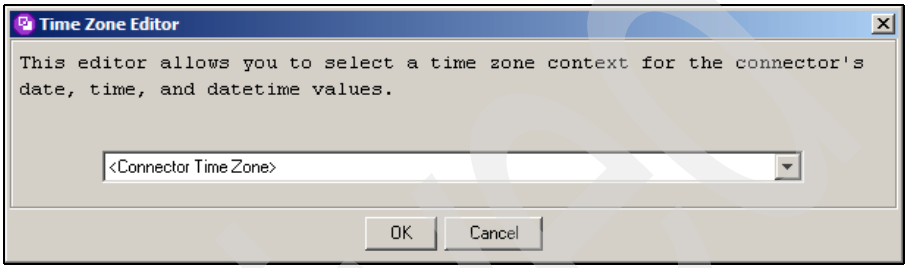



*Figure 7-11 Time Zone Editor window*

- **Disable the Connector**

  The value that is associated with this property determines whether the connector configuration is available for use by clients of Content Integrator. The default value of false indicates that an Content Integrator client application *can* use the connector. A value of true indicates that the connector configuration is disabled and the connector configuration *cannot* be used to connect to a content management system.

- **Logging Information**

  You can associate logging settings with each connector configuration in Content Integrator. Select **Logging Information** to use the Logging Information Editor, as presented in Figure 7-12 on page 335.

Archived

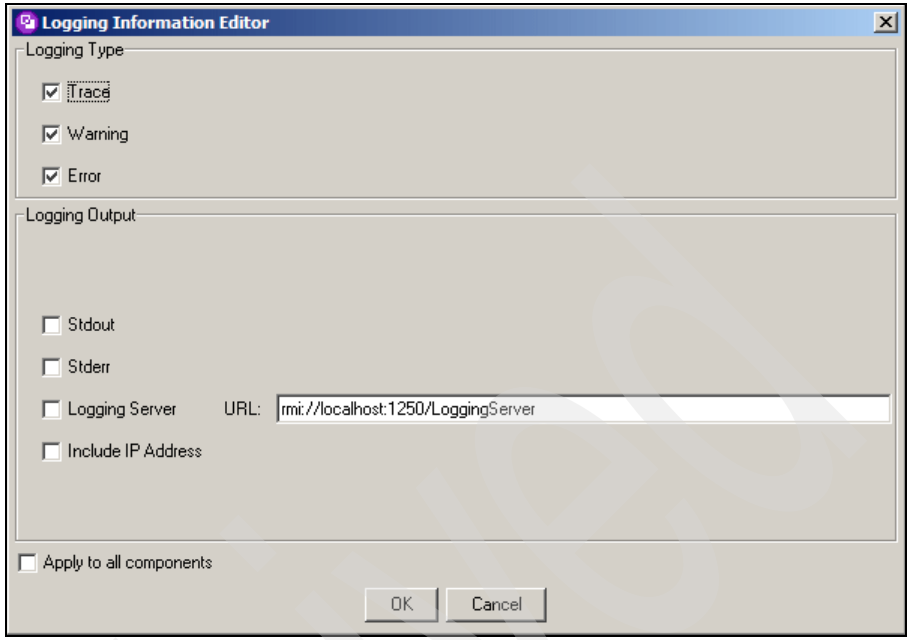



*Figure 7-12 Logging Information Editor window*

Using the Logging Information Editor, you can set the level of logging desired for the connector, as well as the desired console to send the logging output. If you select to log to a logging server, provide a valid logging server URL.

**Trace logging:** Trace logging is intended for debug purposes only. Depending on your environment resources, performance might be severely degraded if trace logging is used.

- **Custom Properties**

Custom connector properties provide a way to pass implementation-specific name-value pairs that can be used by the connector. Typically, custom connector properties are added to alter a specific aspect of the connector operation. To use the Custom Properties Editor that is presented in Figure 7-13 on page 336, select **Custom Properties**. Use the Custom Properties Editor to add, edit, or delete custom properties.

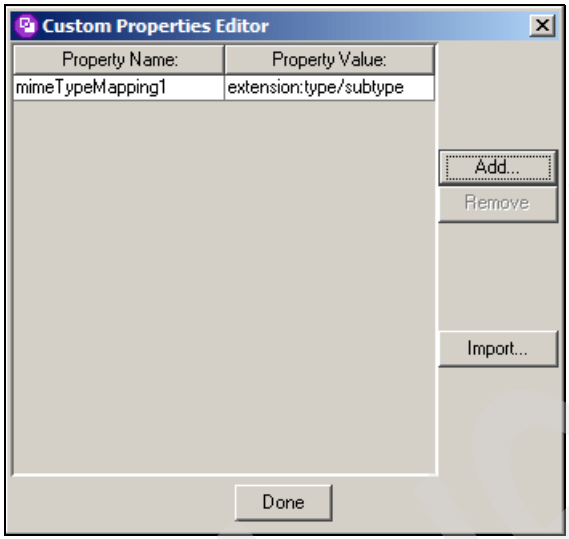



*Figure 7-13 Custom Properties Editor window*

Custom connector properties are specific to connector implementations; a property that is used for one connector implementation likely is not a valid property for another connector implementation. For information about the set of custom connector properties available for each connector implementation, see the appropriate discussion in this chapter or the Content Integrator Information Center.

## 7.3.1 EMC Documentum connector configuration

The EMC Documentum connector enables Content Integrator access to content that is stored in an EMC Documentum content management system. The Documentum connector uses the EMC Documentum Foundation Classes (DFC) Java API. We provide configuration details for DFC Version 6.5, Version 6.0, and Version 5.3.x.

### Connector environment setup

You must install and configure the EMC Documentum Foundation Classes, following the DFC installation documentation, on the same server that is used to host the Documentum connector. The installation owner of DFC must be the same user running the Documentum connector.

On UNIX systems, insure that the same user that is used to install Content Integrator is the same user that is used to install EMC Documentum Foundation Classes.

You can modify the Documentum docbroker host value and the maximum session count value in the DFC configuration file. See the EMC Documentum documentation for details.

To configure the Documentum connector:

1. Install EMC Documentum Foundation Classes (DFC) on the system where the Content Integrator Documentum connector is hosted.
2. Configure the classpath of the Java virtual machine (JVM) hosting the Documentum connector to include two additional items:
   - The path to the `dctm.jar` file
   - The path to the DFC configuration directory

   See the DFC installation guide for information about how to locate these paths. The configuration directory contains the `dfc.properties` file.

   For example, if you are running the connector with the RMI proxy connector, add the DFC configuration directory to the classpath that is specified in the *IICE_HOME*/bin/RMIBRidge.bat or *IICE_HOME*/bin/RMIBridge.sh file, where *IICE_HOME* is the installation directory of Content Integrator. The classpath is typically configured using the *VBR_ALLJARS* variable in the `RMIBridge.bat` or the `RMIBridge.sh` file:

   ```
   set
   VBR_ALLJARS=%VBR_ALLJARS%;C:/Progra~1/Documentum/dctm.jar;C:/Documen
   tum/config
   ```

   **Tip:** If using EMC Documentum Foundation Classes Version 5.3.*x*, use the `dmcl.ini` configuration file instead of the `dfc.properties` file.

3. Set the connector properties using the Administration Tool.

## Connector properties

You configure the Documentum connector-specific properties, which are presented in Figure 7-14 on page 338, using the Administration Tool.

Archived

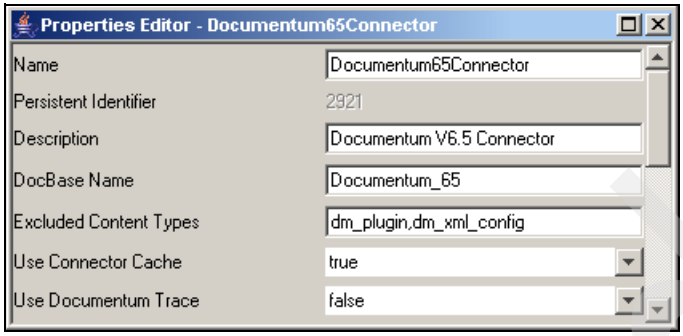



*Figure 7-14 Properties Editor for the Documentum Connector*

We list the properties and their definitions:

- **Docbase Name**

  The EMC Documentum docbase name is case sensitive.

  > **Tip:** If using EMC Documentum Foundation Classes Version 5.3.*x*, the docbase name takes the form DocBase@DocBroker. If DocBroker is omitted, the DFC API will search the `dmcl.ini` configuration file for the DOCBROKER_PRIMARY specification and will attempt to connect using those values.

- **Excluded Content Types**

  This property is a comma-delimited list of content classes to exclude from use when creating content in EMC Documentum.

- **Use Connector Cache**

  This property is an optional flag that specifies whether to use caching for the connector. The default value is true. When the cache is enabled, the connector will cache content class information. No document caching is performed.

- **Use Documentum Trace**

  This property is an optional flag that specifies whether to use DFC tracing. The default value is false. When tracing is enabled, performance is severely degraded. Only enable this trace when debugging is an issue.

- **LockDownPrincipal** (**custom property**)

  This field is the user name or group name to use for the document owner after a document is locked. For more information about the use of this property, see “Locking and unlocking content” on page 340.

Archived

- **USE_CHRONICLE_ID** (custom property)

  For content items with multiple versions, the Documentum connector might return a separate item ID for each version. Applications that rely on all of the versions within a version series to have the same item ID will not function properly without setting this custom property to true.

  Prior to Content Integrator Version 8.4, the Documentum connector based the item ID on the r_object_id property of a Documentum content item. The r_object_id value varies from version to version of a single content item's version series. To use r_object_id for the item ID value, set the custom property USE_CHRONICLE_ID to false.

  Now, Content Integrator can return an item ID based on either the r_object_id or the i_chronicle_id property of Documentum content items. The i_chronicle_id property value remains the same from version to version of a single content item's version series. To use i_chronicle_id for the item ID value, set the custom property USE_CHRONICLE_ID to true.

  The default value of the USE_CHRONICLE_ID custom property is set to true when a Documentum connector configuration is created.

## Additional connector information

Next, we list additional information about the Documentum connector:

- **Documentum security**

  The Documentum connector supports the advanced, enhanced, and the read-token security models in Content Integrator. Table 7-1 presents the logical mapping between EMC Documentum and Content Integrator advanced security model permissions.

*Table 7-1 Mapping between EMC Documentum and Content Integrator permissions*

| Documentum basic security | CanDelete permission | CanUpdate permission | CanRead permission |
|---|---|---|---|
| DELETE | True | True | True |
| WRITE | False | True | True |
| READ (RELATE, VERSION) | False | False | True |
| NONE (BROWSE) | False | False | False |

The EMC Documentum extended permission Change Permission maps to the Content Integrator permission CanChangeSecurity.

Documentum repository access control lists (ACLs) can be categorized in domains; however, Content Integrator allows only one call using the operation ContentBase.replaceAccessControlList(String). This class accepts the ACL name only. You can pass a combined string with the ACL name and ACL domain using ContentBase.replaceAccessControlList(String) using the format aclName[_]aclDomain. The Documentum connector separates the two strings into the ACL name and domain, using an underscore (_) as the delimiter. If the parameter does not have the underscore, the entire string is used as the ACL name.

- **Locking and unlocking content**

  You can configure the Documentum connector to enable Content Integrator to lock and unlock content stored in EMC Documentum. Using the Administration Tool, define the LockDownPrincipal custom property. The LockDownPrincipal property is the name of the user or group that is the content owner after content items are locked.

  The lockDownRecord() operation modifies the security of content so users and groups in the ACL have at the highest, the Version privilege, and it grants the Delete and Change Security privileges to the content item's owner. The operation also changes the content item's owner to the user or group that is specified in the LockDownPrincipal custom property.

  The value specified in the LockDownPrincipal custom property must be the name of the user that will call the lockDownRecord() operation or the name of a group of which the user is a member. The user name or group name must be valid in the ECM Documentum system.

- **Virtual documents**

  The Documentum connector supports virtual documents, representing them as multipart content in Content Integrator.

## 7.3.2 Hummingbird Document Management connector configuration

The Hummingbird Document Management connector enables Content Integrator to access content that is stored in a Hummingbird Document Management content management system. The Hummingbird Document Management connector uses the Hummingbird Document Management DCOM services API. We provide configuration details for Hummingbird Document Management Version 6 and Hummingbird Document Management Version 5.

### Connector environment setup

You can easily configure the Hummingbird Document Management connector with or without the use of the RMI proxy connector.

**Hummingbird Document Management**

To configure the Hummingbird Document Management connector:

1. Create a new user.

   The Hummingbird Document Management connector includes native components that must run as part of the operating system. To simplify the configuration, create a user specifically for these native components and give that user permission to act as part of the operating system:

   a. From Windows, access the Computer Management tool by selecting **Start** → **Administrative Tools** → **Computer Management**.

   b. Within Computer Management, select **Computer Management (local)** → **System Tools** → **Local Users and Groups** → **Users** from the tree view.

   c. Right-click the **Users** folder, and select **New User.**

   d. Specify a new user name, SysAdmin, and a password that never expires.

   e. Click **Create**.

2. Register the native components.

   The Hummingbird Document Management connector requires access to native components. In this task, we register the appropriate dynamic link library (DLL) to allow access to the native components:

   a. Copy the *ICI_HOME*/combridge/bin/setdllhost.exe file and the entire *ICI_HOME*/combridge/bin/international directory to the *HB_HOME*/DM API directory where *ICI_HOME* is the installation directory of Content Integrator and *HB_HOME* is the installation directory of Hummingbird Document Management.

   b. On the Hummingbird Document Management machine, run the following command from the *HB_HOME*/DM API directory:

   ```
   setdllhost.exe Hummingbird DM installation folder/DM
   API/PCDClient.dll "PCDClient"
   ```

   c. Restart the Hummingbird Document Management system.

3. Set the connector properties using the Administration Tool.

Archived

## Connector properties

We configure the Hummingbird Document Management connector-specific properties, as presented in Figure 7-15, using the Administration Tool.

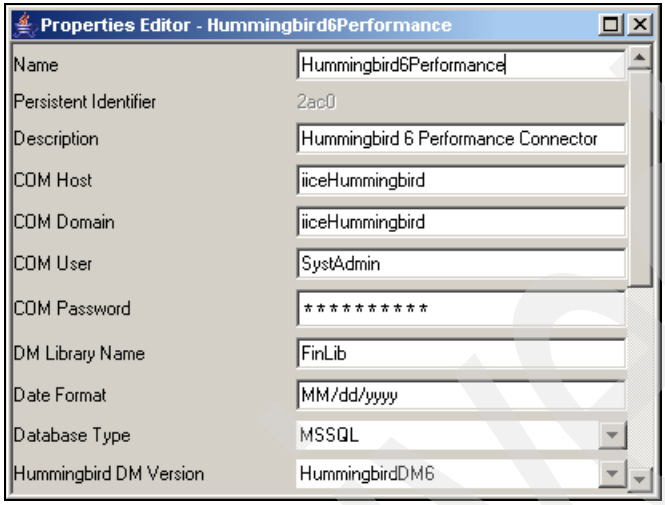



*Figure 7-15 Properties Editor for the Hummingbird Document Management connector*

We list the properties and their definitions:

- **COM Host**

  This property is the host name or IP address of the remote COM server. For a connector that is deployed on the same server as the COM API, you can leave this property blank or set it to `localhost`. This setting must be consistent within a JVM.

- **COM Domain**

  This property is the domain name for user authentication on the remote COM server. Provide a value even if the connector is deployed on the same server as the COM API.

- **COM User**

  This property is the user name for user authentication on the remote COM server. For a connector that is deployed on the same server as the COM API, you typically leave this property blank. Leaving this value blank causes the Java-COM connector to use a native call to retrieve the user name. This valid user name is the user that must have Act as part of the operating system policy that is assigned to it.

Archived

- **COM Password**

  This property is the password for user authentication on the remote COM server.

- **DM Library Name**

  This property is the name of the Hummingbird Document Management library.

- **Date Format**

  This property is the date format that is used by the Hummingbird Document Management API when returning date and DateTime values.

- **Database Type**

  This property is the type of database that holds the Hummingbird Document Management Library. The three options for this property are MSSQL, Oracle, or Sybase.

- **Hummingbird DM Version**

  This property is the version of your Hummingbird Document Management environment. The two options for this property are HummingbirdDM5 or HummingbirdDM6.

### Additional connector information

Regarding the memory usage and the Hummingbird Document Management connector when retrieving many documents, the memory used by the Hummingbird client continues to increase as each native content item is retrieved. If the memory usage reaches the maximum allocated memory size, performance might be affected. You can change the value of the Maximum Total Size (KB) setting for the Hummingbird repository, which specifies the total amount of memory that can be used for caching. The default value for the cache size is 2,000,000 KB. If you have performance issues with the Hummingbird connector, change the value to 200,000 KB.

The Hummingbird Document Management connector supports native content retrieval and upload using streams.

## 7.3.3 IBM Content Manager connector configuration

The Content Manager connector enables Content Integrator access to content that is stored in an IBM Content Manager content management system. The Content Manager connector uses the IBM DB2 Information Integrator for Content API. We provide configuration details for IBM Content Manager Version 8.4 and IBM Content Manager Version 8.3.

Archived

## Connector environment setup

Content Integrator Version 8.5.1 supports both IBM Content Manager Version 8.3 and Version 8.4. Because of the support of these two versions, the connector configuration can differ depending on the details of your environment.

If you run IBM Content Manager Version 8.3 and require the type 2 Java Database Connectivity (JDBC) driver for DB2, the DB2 runtime client is required to allow Content Integrator to be deployed separately from IBM Content Manager.

**IBM Content Manager Version 8.4 with and without an RMI proxy connector**
IBM Content Manager V8.4 now supports both the type 4 and type 2 DB2 drivers for JDBC and SQLJ. The configuration of the connector differs depending on the driver type that you select.

> **Type 2 driver:** If your environment requires the type 2 driver, refer to the steps that are provided for IBM Content Manager Version 8.3.

To configure the Content Manager connector for IBM Content Manager Version 8.4 using the DB2 driver for JDBC and SQLJ type 4, perform the following steps:

1. Locate the `cmbicmsrvs.ini` file, which is usually found in the DB2 Information Integrator for Content (II4C) installation directory:

   ```
   II4C_ROOT/cmgmt/connectors
   ```

2. In the `cmbicmsrvs.ini` file, set the values for the host name, the port, and the database name:

   ```
   ICMHOSTNAME=myhostname.ibm.com
   ICMPORT=50000
   ICMREMOTEDB=icmnlsdb
   ICMJDBCDRIVER=com.ibm.db2.jcc.DB2Driver
   ICMJDBCURL=
   ```

3. If you run an RMI proxy connector on a system that is hosted in an environment *other than* the Content Integrator installation, install Content Integrator on that machine (Otherwise, skip to step 4):

   a. Insure that the RMI proxy connector uses IBM Java Runtime Environment (JRE) Version 1.5, which is required by IBM Content Manager.

   b. Modify the RMI proxy connector classpath to point to the appropriate II4C files. If your RMI proxy connector runs on Windows, add the following lines to your `RMIBridge.bat` file before the line `set VBR_RMIPORT=1251`:

      ```
      set IBMCMROOT=C:/Program Files/IBM/db2cmv8
      set CM_JARS=%IBMCMROOT%/lib/cmbsdk81.jar
      set CM_JARS=%IBMCMROOT%/lib/cmbview81.jar;%CM_JARS%
      ```

```
set CM_JARS=%IBMCMROOT%/lib/cmb81.jar;%CM_JARS%
set CM_JARS=%IBMCMROOT%/lib/db2jcc.jar;%CM_JARS%
set CM_JARS=%IBMCMROOT%/lib/db2jcc_license_cu.jar;%CM_JARS%
set CM_JARS=%IBMCMROOT%/lib/db2jcc_license_cisuz.jar;%CM_JARS%
set CM_JARS=%IBMCMROOT%/cmgmt;%CM_JARS%
set VBR_ALLJARS=%VBR_ALLJARS%;%CM_JARS%
```

If your RMI proxy connector runs in a Linux/UNIX environment, add the following lines to your `RMIBridge.sh` file:

```
IBMCMROOT=/opt/IBM/db2cmv8
export IBMCMROOT
CM_JARS=$IBMCMROOT/lib/cmbsdk81.jar
export CM_JARS
CM_JARS=$IBMCMROOT/lib/cmbview81.jar:$CM_JARS
export CM_JARS
CM_JARS=$IBMCMROOT/lib/cmb81.jar:$CM_JARS
export CM_JARS
CM_JARS=$IBMCMROOT/lib/db2jcc.jar:$CM_JARS
export CM_JARS
CM_JARS=$IBMCMROOT/lib/db2jcc_license_cu.jar:$CM_JARS
export CM_JARS
CM_JARS=$IBMCMROOT/lib/db2jcc_license_cisuz.jar:$CM_JARS
export CM_JARS
CM_JARS=$IBMCMROOT/cmgmt:$CM_JARS
export CM_JARS
VBR_ALLJARS=$VBR_ALLJARS:$CM_JARS
```

**IBM Content Manager Version 8.3:** Instead of the `db2jcc.jar` file and the `db2jcc_license_cu.jar` file, include the `db2java.zip` file, which is usually found in the `C:/sqllib/java` directory on Windows or the `/opt/sqllib/java` directory on Linux/UNIX.

c. Proceed to step 5.

4. If you do *not* use an RMI proxy connector, you must add the appropriate files to the Content Integrator classpath. For example, copy the following files to the *ICI_HOME*`/lib` classpath:

```
II4C_ROOT/lib/cmb81.jar
II4C_ROOT/lib/cmbview81.jar
II4C_ROOT/lib/db2jcc.jar
II4C_ROOT/lib/db2jcc_license_cu.jar
II4C_ROOT/cmgmt/connectors/cmbicmsrvs.ini
II4C_ROOT/cmgmt/connectors/cmbicmenv.ini
```

Archived

> **Note:** An additional JAR file, `db2jcc_license_cisuz.jar`, is required to connect to System z servers. For Content Manager Version 8.3, include the `db2java.zip` file instead of the previously mentioned JAR files. You can find the `db2java.zip` file usually in the `C:/sqllib/java` directory on Windows or the `/opt/sqllib/java` directory on Linux/UNIX.

5. Set the connector properties using the Administration Tool.

**IBM Content Manager Version 8.3 without an RMI proxy connector**
To configure the Content Manager connector for IBM Content Manager Version 8.3, without an RMI proxy connector, perform the following steps:

> **Note:** If you run the RMI proxy connector on your IBM Content Manager machine, refer to the configuration instructions for IBM Content Manager Version 8.4 in the previous section. Running RMI on your IBM Content Manager server avoids the need to install the database client and DB2 Information Integrator for Content.

1. Install the database client to connect to the database that is used by IBM Content Manager. For example, install the DB2 runtime client if DB2 is your database server.
2. To configure the DB2 runtime client, open the DB2 command window (**Start** → **Programs** → **IBM DB2** → **Command Line Tools** → **Command window**) and execute the following commands:
   a. `db2 catalog tcpip node NAME remote HOSTNAME server PORTNUMBER`
   where:
      - *NAME* can be set to any name.
      - *HOSTNAME* must be the host name or the IP address for the IBM Content Manager server.
      - *PORTNUMBER* is the port where DB2 runs on the IBM Content Manager server. The default port number where DB2 runs is 50000.

   b. `db2 catalog db DBNAME as DBALIAS at node NAME`
   where:
      - *NAME* is the name for the node in the previous command example.
      - *DBNAME* is the database name on the IBM Content Manager server.
      - *DBALIAS* is any name that you want to call the database.
3. Test the connection to DB2 database with this command:

   `db2 connect to DBALIAS user USERNAME using PASSWORD`

4. Proceed with the steps for configuring the Content Manager connector for IBM Content Manager Version 8.4 in the previous section.
5. Set the connector properties using the Administration Tool.

## Connector properties

The Content Manager connector-specific properties that are presented in Figure 7-16 are configured using the Administration Tool.

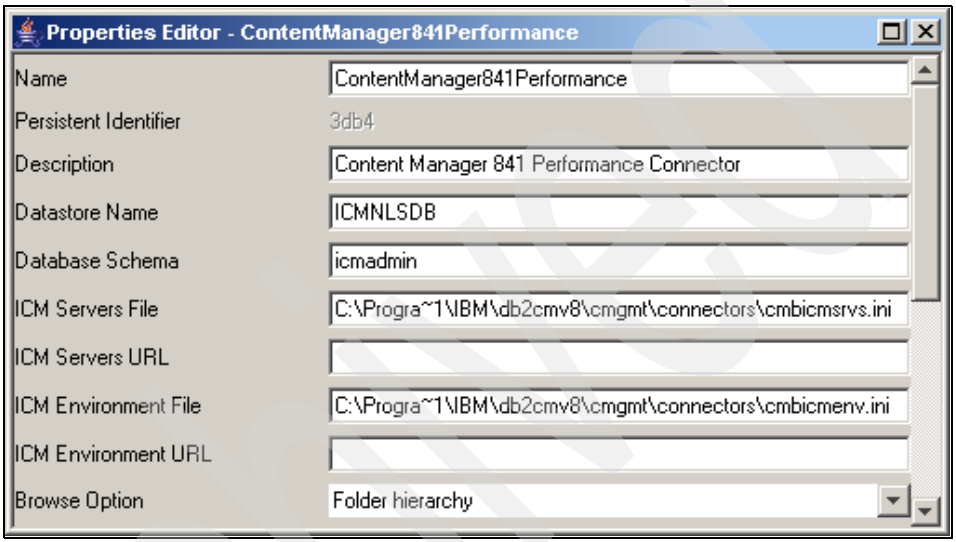



*Figure 7-16 Properties Editor for the Content Manager connector*

We list the properties and their definitions:

- **Datastore Name**

  This property is the name of the IBM Content Manager data store to which to connect.

- **Database Schema**

  This property is the name of the database schema. If the Database Schema property is not set, the IBM Content Manager data store gathers the schema name from the INI file that is specified in the ICM Servers File or the ICM Servers URL property.

- **ICM Servers File**

  This property specifies the file location of the `cmbicmsrvs.ini` file. This INI file stores the data source information for the data store.

- **ICM Servers URL**

  This property specifies the URL location of the data source information. You do not need to set this property if the ICM Servers File property is set.

- **ICM Environment File**

  This property specifies the location of the `cmbicmenv.ini` file. This INI file stores the database connect information.

- **ICM Environment URL**

  This property specifies the URL location of the database connection information. You do not need to set this property if the ICM Environment File property is set.

- **Browse Option**

  This property specifies how a client displays content objects for browsing. This property has two valid values:

  – Folder_Hierarchy: Displays the default root folder browsing view, which lists all first-level folders but no content.

  – No_Hierarchy: Displays the root folder but no child folders. Users must locate folders or content by searching.

- **Maximum-Foreignkey-Records** (custom property)

  If a foreign key exists for a property, it gets the list of foreign key values and assigns them as available values. This custom property enables you to set the limit of foreign key values:

  – The default, 0, retrieves all of the foreign key values.

  – Any number > 0 retrieves the specified number of foreign key values.

  – Specifying -1 or < -1 does not retrieve any foreign key values.

- **mimeTypeMapping*xx*** (custom property)

  This property enables you to map file extensions to Multipurpose Internet Mail Extensions (MIME) types that can be selected when you configure native content, for example:

  ```
  mimeTypeMapping1=jpg:image/jpeg
  ```

- **DBAUTH** (custom property: data store configuration)

  This property houses the database authentication information. Valid values are SERVER and CLIENT. If SERVER is specified, the database user ID and password are provided to the server. If CLIENT is specified, the database user ID and password are not provided to the server. The default value is SERVER.

Archived

- **LDAPORG** (custom property: data store configuration)

  This property specifies the Lightweight Directory Access Protocol (LDAP) organization that provides the directory information. Use this property when the list of data sources is in IBM Secure Way.

- **CHKDBPOOL** (custom property: data store configuration)

  This property checks for database pooling. Valid values are YES and NO. The default is YES.

- **TitleProperty-*xxxxx*** (custom property: data store configuration)

  This property provides a field to use as the folder name for a specific item class, for example, the value of the property name TitleProperty-XYZ_InsPolicy might be XYZ_AdjustFName.

- **SQLUID** (custom property: data store configuration)

  This property specifies the user ID for the database.

- **SQLPWD** (custom property: data store configuration)

  This property specifies the user's password for the database.

- **SSO** (custom property: data store configuration)

  This property specifies single sign-on (SSO) information. Valid values are TRUE and FALSE. If SSO is set to TRUE, DBAUTH must be set to CLIENT, and no user name and no authentication are provided to the connect method.

- **APPLNAME** (custom property: data store configuration)

  This property specifies the name of the application that you are running. You use this property to identify trace and log information.

- **TRACESERVERLEVEL** (custom property: data store configuration)

  This property specifies the trace level for the server. DB2 Content Manager trace flags include these trace flags:

```
0: ICM TRACE OFF
1: ICM TRACE BASIC
2: ICM TRACE DETAILED
4: ICM TRACE DATA
8: ICM TRACE PERFORMANCE
16: ICM TRACE PARSEBUILD
32: ICM TRACE MEMDEBUG
256: ICM TRACE CACHE TRACE
512: ICM TRACE CACHE ALLOC
1024: ICM TRACE CACHE MGT
```

Archived

The trace level is the sum of the numbers that are associated with each trace flag. For example, a common tracing level is 15, which is 1 (ICMTRACE_BASIC) plus 2 (ICMTRACE_DETAILED) plus 4 (ICMTRACE_DATA) plus 8 (ICMTRACE_PERFORMANCE).

- **UsageMode** (custom property)

  You use usage mode to alter what is returned by the connector when a request for retrieving item classes is made through the Content Integrator API. If set to “Search and Retrieval”, the connector filters out IBM Content Manager item classes when the new version policy for parts is set to Never Version or Application Choice. When set to “Records Management”, the connector filters out item classes when the new version policy for the attributes is set to Always create or Application choice.

- **SHIFT_SYSTEM_ATTRIBUTE_TIME_STAMPS** (custom property)

  This property shifts IBM Content Manager system property values of type TIMESTAMP to Greenwich mean time (GMT). The default value is true.

- **includePageCountInQuery** (custom property)

  When set to true, an additional column is added to search request results that identifies the page count for each result.

- **includeURLsInFinder** (custom property)

  When set to true, two additional columns are added to Item Finder results: one column contains an array of URLs (each page) and the other column contains an array of mime types. The URLs are native content management system URLs that can be used to retrieve native content directly from the IBM Content Manager’s resource manager using HTTP.

- **ReadSecurity** (custom property)

  When set to false, the security-related retrieval operations of the Advanced Security model no longer return security information. The Enhanced Security model is not affected. The default value is true.

## Additional connector information

Consider this additional information about the Content Manager connector:

- **Deriving a repository item name**

  Within IBM Content Manager, the item name is not automatically populated. An item can be a folder name, a document name, and so on, so Content Integrator derives a repository item name from IBM Content Manager by applying the following logic:

  – **For folder name**:

    i. TitleProperty-*xxxxx* custom property, if set for this connector

ii. Represents Item attribute from IBM Content Manager
iii. ItemID (an 18-digit number, such as `A05B15B44754D59295`)

– **For content name**:

i. Resource name from IBM Content Manager
ii. TitleProperty-*xxxxx* custom property, if set for this connector
iii. RepresentsItem attribute from IBM Content Manager
iv. Content file name
v. ItemID (an 18-digit number, such as `A05B15B44754D59295`)

– **For work item name**:

i. The name of the attached item
ii. If the attached item is a folder, follow the logic to get a folder name.
iii. If the attached item is content, follow the logic to get a content name.

► **Search requests using system-defined properties**

When an application creates an item in IBM Content Manager, several system-defined attributes are assigned to the item. You can construct searches that use any of these system-defined attributes, as identified in Table 7-2.

*Table 7-2 IBM Content Manager system-defined attributes*

| Attribute name | Data type | Description |
|---|---|---|
| CREATEUSERID | String | Creator user ID |
| CREATETS | DateTime | Creation time stamp |
| LASTCHANGEDUSERID | String | Last changed user ID |
| LASTCHANGEDTS | DateTime | Last changed time stamp |
| ACLCODE | Integer | Access control list code |
| COMPONENTID | String | Component ID |
| ITEMID | String | Item ID |
| VERSIONID | Short | Version ID |
| EXPIRATIONDATE | DateTime | Expiration Date |

► **Content Manager security**

The Content Manager connector supports the enhanced security model in Content Integrator. The enhanced security model exposes security features of the content management system, such as the security schema, hierarchical groups, access control lists (ACLs), and system privilege types.

Archived

- **Content annotations**

  *Content annotation* enables users to add information to content, such as text, highlights, or shapes. You can view and manipulate content items by using the Content Integrator viewer applet instead of by using the viewing capabilities of the native repository. To add annotations by using the viewer applet, select the action, and then, drag the action to the area of the viewed content to be annotated.

  IBM Content Manager requires that you check out content first before you annotate it.

  Table 7-3 maps the IBM Content Manager annotation types to the corresponding Content Integrator annotation types.

*Table 7-3 Annotation mapping*

| Content Manager annotation type | Content Integrator annotation type |
|---|---|
| Arrow | ARROW |
| Circle | ELLIPSE |
| Pen | FREEHAND |
| Highlight | HIGHLIGHT |
| Stamp | STAMP |
| Line | LINE |
| Note | NOTE |
| Rectangle | RECTANGLE |
| Text | TEXT |

  When text annotations that include non-English characters are added to IBM Content Manager content items, the non-English characters are returned from the repository in an unusable format and appear as question marks (?). Insure that the locale of the computer that runs the viewer applet and the locale of the computer that runs the IBM Content Manager connector are set to the language of the non-English characters that are to be added. The locale of the computer that runs Content Integrator does not need to be changed.

  The Content Integrator annotation type IMAGE_OVERLAY is not supported by the connector.

Archived

- **Content routing**

  The Content Manager connector supports workflow through document routing. Table 7-4 maps IBM Content Manager document routing concepts to content integration workflow objects and properties.

*Table 7-4 Document routing concepts*

| Document routing | Content integration | Description |
|---|---|---|
| Data store | Repository | A data store represents the IBM Content Manager repository. A content integration user session can connect to a single data store. |
| Process | Item class | A process can be initiated, suspended, resumed, completed, and so on. A routing process consists of the defined routes that a package can follow. Multiple routing processes can use the same nodes and can use multiple routes between nodes. |
| Work list | Work queue | A work list filters work packages that are associated with specified work nodes. A work list can filter the work packages that are available to various users. |
| Work package | Work item | A work package is the vehicle through which an item moves in the routing process. A work package contains all of the necessary information about the process and the item that it transports. |
| Item | Internal attachment | An item is a document or folder. Any document or folder can be routed between nodes. Folders can be useful, because they can contain any number of items of any type that can be added or removed as the process progresses. |

You can view the flow of work items through various work queues in a workflow by logging on to the IBM Content Manager System Administration Tool and clicking **Document Routing** → **Processes** → **Process Name** → **Properties**.

Archived

To complete a work item and move it to another work queue, specify information for the COMPLETE_ROUTE_SELECTION property and the COMPLETE_NEXTOWNER property. COMPLETE_ROUTE_SELECTION and COMPLETE_NEXTOWNER are required properties for completing a work item. You can edit these properties, but they are only relevant for completing the work item. These properties are not saved when you click Save changes for the work item.

When you save a work item, the system derives the name of the work item from the folder or content that is attached to the work item.

After you complete a work item that is at the last node of a workflow process, the work item no longer is displayed.

You cannot retrieve a work item's history with this connector.

- **Workflow search**

  You can run content integration queries on an IBM Content Manager workflow. IBM Content Manager work lists are displayed in the swing.RepoBrowser example. However, in workflow queries, IBM Content Manager work nodes are displayed as work queues.

  You can run a query without a workflow or work queue name. Example 7-1 returns work items that have car documents as attachments for all cars of the make Car Company A that are not suspended.

*Example 7-1 Query that returns work items without a workflow or work queue name*

```
ExecQuery DB2CM userid password selectionProperties="MAKE,MODEL,VIN"
qtype=work selectionCriteria="MAKE='CarCompanyA' AND SUSPENDFLAG = 0
AND ATTACHMENT_ITEMTYPE='Car'"
```

You can run a query with the workflow name specified. Example 7-2 returns work items for the Accident Investigation workflow that have car documents as attachments for all cars of the make Car Company A.

*Example 7-2 Work item query with a workflow name specified*

```
ExecQuery DB2CM userid password selectionProperties="MAKE,MODEL,VIN"
qtype=work class=AccidentInvestigation
selectionCriteria="MAKE='CarCompanyA' AND ATTACHMENT_ITEMTYPE='Car'"
```

You can run a query with both workflow and work queue names specified. Example 7-3 on page 355 returns work items for the Accident Investigation workflow, which are at the work queue UnderReview and have car documents as attachments for all cars of the make Car Company A.

*Example 7-3 Work item query with both workflow and work queue specified*

```
ExecQuery DB2CM userid password selectionProperties="MAKE,MODEL,VIN"
queue=UnderReview qtype=work class=AccidentInvestigation
selectionCriteria="MAKE='CarCompanyA' AND ATTACHMENT_ITEMTYPE='Car'"
```

The IBM Content Manager connector supports queries against either content or workflow, but not both content and workflow in a single query. The ALL_QUERY query type is treated as the CONTENT_QUERY query type in the IBM Content Manager connector. To run a workflow query, you must specify the WORKITEM_QUERY query type.

- **Locking and unlocking content**

  You can configure the IBM Content Manager connector to enable Content Integrator to lock and unlock documents that are stored in an IBM Content Manager repository.

  To enable the IBM Content Manager connector to lock and unlock documents that are stored in an IBM Content Manager Version 8.4 repository, you must enable the Content Manager Version 8.4 for records management through IBM Enterprise Records (formerly known as IBM FileNet Records Manager) or IBM Records Manager. See the documentation in IBM FileNet or IBM Content Manager Version 8.4 for detailed instructions about records management enablement. After IBM Content Manager is enabled for records management, any user with the ItemRecordsAdmin privilege can lock and unlock the content. Users who have the AllPriv privilege set, such as icmadmin, can lock and unlock documents.

  You can lock and unlock documents through the lockDownRecord() operation and undeclare a record through the unLockRecord() operation. When a document is locked, all users, including the icmadmin super user, cannot check in the native content. Any user with the AllPriv privilege set can unlock the content even if it was locked by the super user.

  Locking a document and unlocking a document are enabled at the content object level only, and they are supported only by IBM Content Manager Version 8.4.

- **Version delete**

  You can use the IBM Content Manager connector to enable Content Integrator to delete specific versions of documents that are stored in an IBM Content Manager Version 8.4 repository.

  You can use the Content Manager connector to delete specific versions of documents through the deleteVersion() operation. If a document has multiple versions, a user can lock a particular version of the document, such as Version 1.0. Later, that document can be unlocked by the same user. Finally, Version 1.0 can be deleted if it is obsolete or no longer useful to the user.

The steps to delete a version of a document are similar to those steps for deleting a document. An administrator can delete a version in the tool of the repository, which Content Integrator calls the deleteVersion() operation.

An administrator can delete a version of a content item by using the Repository.deleteVersion operation in the Content Integrator Java API. The DeleteVersion.java command-line sample is provided as an example of how to use this feature. See the Content Integrator API documentation in the *ICI_HOME*/docs/integrate/API/ directory, where *ICI_HOME* is the Content Integrator installation directory.

IBM Content Manager Version 8.4 or later is required for the version delete capability.

- **Mutable content**

  IBM FileNet P8 Content Federation Services (CFS) does not support mutable content, but IBM Content Manager does. IBM Content Manager allows an item to be updated without creating a new version if the version rule is set to Never or Application. Therefore, CFS uses the custom property UsageMode to make the Content Manager Connector filter out all item classes that allow mutable content. CFS sets the custom property to the “Search and Retrieve” value during the crawling phase to filter out item classes that have mutable native content. CFS sets the custom property to the “Records management” value during lockdown to filter out item classes with mutable metadata.

## 7.3.4 IBM FileNet Content Services connector configuration

The IBM FileNet Content Services connector enables Content Integrator access to content that is stored in an IBM FileNet Content Services content management system. The IBM FileNet Content Services connector uses the IBM FileNet Integrated Desktop Management and Web Services Toolkit API, which is also known as the IBM FileNet Integrated Desktop Management Web Client. We provide the configuration details for IBM FileNet Content Services Version 5.5 and Version 5.4.

### Connector environment setup

The setup and configuration of the IBM FileNet Content Services connector requires only a few steps to complete. The only requirement is that the connector JVM runs on the same server where IBM FileNet Integrated Desktop Management and Web Services Toolkit have been installed.

**Requirement:** IBM FileNet Integrated Desktop Management and Web Services Toolkit require a Windows-based environment. If your Content Integrator installation is on a UNIX platform, you are required to use an RMI proxy connector, because the IBM FileNet Content Services connector must be colocated with the Toolkit.

To configure the IBM FileNet Content Services connector:

1. Using the DCOM configuration utility, insure that the IBM FileNet Integrated Desktop Management (IDM) Find component is associated with a user that has sufficient Windows privileges to run the FileNet IDM Find component:
   a. On the server where IBM FileNet Integrated Desktop Management and Web Services Toolkit run, select **Start** → **Run.**
   b. Enter the **`dcomcnfg`** command, and select **OK**.
   c. Locate the FileNet IDM Find component and view its properties.
   d. Verify that the Identity settings specify credentials for a user that has sufficient Windows privileges to run the FileNet IDM Find component.
   e. If changes are made, restart the Windows server.
2. Register the PanagonWorker4.dll component:
   a. On the server where IBM FileNet Integrated Desktop Management and Web Services Toolkit run, select **Start** → **Run.**
   b. Type the command:

      ```
      regsvr32 ICI_HOME/bridges/panagon/native/PanagonWorker4.dll
      ```

      where *`ICI_HOME`* is your Content Integrator installation directory. Select **OK**.
3. Set the connector properties using the Administration Tool.

Archived

## Connector properties

The IBM FileNet Content Services connector-specific properties that are presented in Figure 7-17 are configured using the Administration Tool.

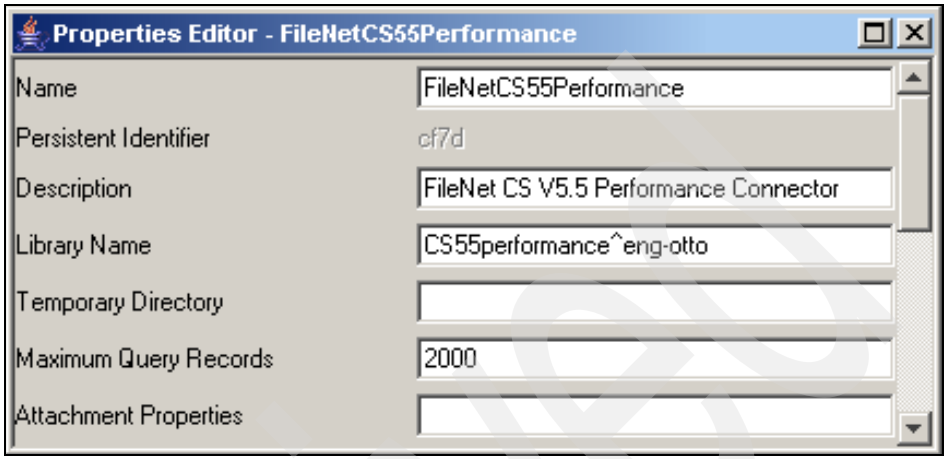



*Figure 7-17 Properties Editor for the IBM FileNet Content Services connector*

We list the properties and their definitions:

- **Library Name**

  This property is the fully qualified IBM FileNet library name in the form *system^host*. The *system* value is the name of the IBM FileNet Content Services library or system to use. The *host* value is the name of the server where IBM FileNet Content Services is running.

- **Temporary Directory**

  This property is a temporary location where the IBM FileNet Content Services connector temporarily stores content. If specified, the directory must be available and accessible on the server where the IBM FileNet Content Services connector is deployed. To use the default temporary location of *ICI_HOME*/temp, leave this property blank.

- **Maximum Query Records**

  This property is the maximum number of results to return in a search request. The default value is 2000.

- **Attachment Properties**

  This property is a comma-delimited list of field names to be treated as content attachments or references that are external to the IBM FileNet Content Services content management system. Attachments are supported with work items only.

- **LockDownPrincipal** (custom property)

  This property is the user identifier to use as the document owner after a document is locked. For more information about the use of this property, see "Locking and unlocking content" on page 360.

- **MimeType-*xxxxx*** (custom property)

  Requesting MIME types from IBM FileNet Content Services is an expensive operation relative to similar requests in other content management systems. By default, the Content Integrator's FileNet Content Services connector uses an internal mapping of native content file extensions to determine MIME types. You can change this behavior using the MimeType-*xxxxx* custom property.

  To request that the IBM FileNet Content Services connector retrieve MIME Type information from the IBM FileNet Content Services system for a specific MIME type, add the custom property in the format MimeType-*xxxxx* where *xxxxx* is the file name extension to use. For example, to require the system to request MIME type information for the doc file name extension, use MimeType-doc for the custom property name and provide a blank value for the custom property value.

- **systemProperties** (custom property)

  You can request that the IBM FileNet Content Services connector retrieve system properties with content using the systemProperties custom connector property. Table 7-5 presents the three valid uses for this custom property. The default value for this property is NONE.

*Table 7-5 Valid systemProperties settings*

| Action | Description |
|---|---|
| systemProperties = NONE | No system properties are retrieved. This setting is the default setting. |
| systemProperties = ALL | All system properties are retrieved. Note that this option will decrease connector performance. |
| systemProperties = *propertyName1, propertyName2, propertyName3* | Only the specified system property names are retrieved. In this case, *propertyName1, propertyName2,* and *propertyName3* are retrieved. |

Archived

## Additional connector information

Additional information about the IBM FileNet Content Services connector:

- **IBM FileNet Content Services security**

  The IBM FileNet Content Services connector supports the advanced and enhanced security models in Content Integrator. With the advanced security model, application developers can set separate read, update, delete, and change security privileges for users and groups. The enhanced security model exposes security features of the content management system, such as the security schema, hierarchical groups, ACLs, and system privilege types.

- **Locking and unlocking content**

  You can configure the IBM FileNet Content Services connector to enable Content Integrator to lock and unlock content that is stored in IBM FileNet Content Services. Using the Administration Tool, define the LockDownPrincipal custom property. The LockDownPrincipal property is the name of the user or group that becomes the content owner after content items are locked.

  The lockDownRecord() operation modifies the content's permissions to read-only and transfers the content item's owner to the user identifier that is specified in the LockDownPrincipal custom property. The lockDownRecord() operation also grants administrative-level permissions to the content's owner.

  The value that is specified in the LockDownPrincipal custom property must be the name of the user that will call the lockDownRecord() operation or the name of a group of which the user is a member. The user name or group name must be valid in the IBM FileNet Content Services system. It is important that the user have its own access control entry in the content's ACL that grants the required privileges for locking content. The principal cannot lock content if the principal obtains the privileges that are needed for locking a document by being a member of a group that has these privileges.

  Locking content and unlocking content are enabled at the content object level only.

- **Object locking**

  Content Integrator relies on the object locking controls that are provided by IBM FileNet Integrated Desktop Management and Web Services Toolkit to perform operations on content. Other IBM FileNet Content Services clients can use the same Web Services locking controls concurrently with Content Integrator.

- **Version delete**

  You can use the IBM FileNet Content Services connector to delete the specific versions of documents that are stored in an IBM FileNet Content Services content management system. Unlike other systems, however, when

a specific version of content is deleted, the native content for that version is removed while the metadata is retained. IBM FileNet Content Services document version list will display the deleted version as Offline (Non-reclaimable).

See the `DeleteVersion.java` command-line example file for details about how to use this feature.

> **Important:** IBM FileNet Web Services 4.0.2-003 or later is required for version delete capability.

## 7.3.5 IBM FileNet Content Manager connector configuration

The IBM FileNet Content Manager connector enables Content Integrator access to content that is stored in an IBM FileNet Content Manager enterprise content management system. The FileNet Content Manager connector uses the IBM FileNet Version 3.5.*x* Content Java API. We provide configuration details for FileNet Content Manager Version 4.5 running on IBM WebSphere Version 6.1.*x* and for IBM FileNet Content Manager Version 3.5.*x* running on Oracle WebLogic.

### Connector environment setup

Due to recent architectural updates in the FileNet Content Manager product, the setup and configuration of the FileNet Content Manager connector has become quite complex. Therefore, we strongly recommend that you use an RMI proxy connector to host the connector.

Differences in connector configuration vary enough to warrant separate descriptions based on your environment scenario:

- IBM FileNet Content Manager Version 4.5 with the connector hosted on the IBM FileNet Content Manager server
- IBM FileNet Content Manager Version 4.5 with the connector hosted in an environment *other than* the IBM FileNet Content Manager server
- IBM FileNet Content Manager Version 3.5.*x*

Archived

### *IBM FileNet Content Manager Version 4.5 with the connector hosted on the IBM FileNet Content Manager server*

To configure the FileNet Content Manager connector for IBM FileNet Content Manager Version 4.5 running on IBM WebSphere Version 6.1.*x*, with the connector hosted on the IBM FileNet Content Manager server, perform the following steps:

**Requirements:**

- You must have the IBM FileNet Content Manager Application Engine installed to obtain the libraries that are needed to configure the connector.
- If lockdown functionality is required, you must have IBM WebSphere Application Client Version 6.1.0.11 or later installed on the server that hosts the connector. You also need Fix Pack (FP) 42959.

**Variables:** Refer to these variables, which are used in the following instructions:

- *AE_HOME*: The IBM FileNet Content Manager Application Engine installation directory, typically, *P8_HOME*/AE
- *ICI_HOME*: The Content Integrator installation directory where the RMI proxy connector for FileNet Content Manager is running
- *P8_HOME*: The IBM FileNet Content Manager installation directory
- *WAS_HOME*: The IBM WebSphere Application Server installation directory running IBM FileNet Content Manager

1. Copy the following files from the IBM FileNet Content Manager installation to the *ICI_HOME*/lib directory where the FileNet Content Manager RMI proxy connector resides. These files are typically located in the *AE_HOME*/Workplace/WEB-INF directory:

   ```
   /lib/Jace.jar
   /lib/javaapi.jar
   WcmApiConfig.properties
   ```

2. Edit the properties in the WcmApiConfig.properties file to refer to the IBM FileNet Content Manager server. Replace *host* with the host name or IP address where IBM FileNet Content Manager runs. Port 2809 is the application server default; modify it as necessary for your deployment:

   ```
   RemoteServerUrl=cemp:iiop://host:2809/FileNet/Engine
   RemoteServerUploadUrl=cemp:iiop://host:2809/FileNet/Engine
   RemoteServerDownloadUrl=cemp:iiop://host:2809/FileNet/Engine
   ```

3. In the same `WcmApiConfig.properties` file that you modified in step 2, add the following line to specify the key for locating a UserTransaction reference during an IBM FileNet Content Manager Java Naming and Directory Interface (JNDI) lookup:

   ```
   TxJndiKey=jta/usertransaction
   ```

Now that the connector environment has been configured, update the RMI proxy connector script file in the *ICI_HOME*/`bin` directory:

1. Add a *WAS_HOME* variable to the `RMIBridge.bat` or `RMIBridge.sh` file, setting it to the IBM WebSphere Application Server installation directory. For example, if IBM WebSphere Application Server is installed to the directory `C:/Progra~1/IBM/WebSphere/AppServer`, use

   ```
   set WAS_HOME=C:/Progra~1/IBM/WebSphere/AppServer
   ```

2. Optional: Add the two JAR files that you copied to the *ICI_HOME*/`lib` directory in step 1 (`Jace.jar` and `javaapi.jar` files) to the *VBR_ALLJARS* variable:

   ```
   set VBR_ALLJARS=%VBR_ALLJARS%;VBR_HOME/lib/Jace.jar
   set VBR_ALLJARS=%VBR_ALLJARS%;VBR_HOME/lib/javaapi.jar
   ```

   Under most circumstances, these JAR files will be included in the Java classpath by the `config.bat` or `config.sh` file, because all JAR files are picked up from the *ICI_HOME*/`lib` directory, by default. However, if you have modified the `config.bat` or `config.sh` file for any reason for your environment, you must perform this step.

3. Add a *JNDI_CLIENT_PROVIDER* variable. Set this variable to the host name or IP address of the system running the IBM WebSphere Application Server for IBM FileNet Content Manager. Replace *host* with the host name or IP address where IBM FileNet Content Manager is running:

   ```
   set JNDI_CLIENT_PROVIDER=iiop://host:2809
   ```

4. Add a *JNDI_CLIENT_FACTORY* variable. Set this variable to the naming factory that is being used:

   ```
   set
   JNDI_CLIENT_FACTORY=com.ibm.websphere.naming.WsnInitialContextFactor
   y
   ```

5. Set the following Java options to start the RMI proxy connector. These options are added after the **`java`** command and before the **`RMIBridgeFactoryLauncher`** is specified. Add Java options to include the directories in the client:

   ```
   -Djava.ext.dirs=%WAS_HOME%/java/jre/lib/ext;%WAS_HOME%/java/jre/lib;
   %WAS_HOME%/classes;%WAS_HOME%/lib;%WAS_HOME%/lib/ext ^
   ```

6. Add the boot classpath:

   ```
   -Xbootclasspath/p:%WAS_HOME%/java/jre/lib/ibmorb.jar;%WAS_HOME%/
   profiles/myProfile/properties ^
   ```

   where *myProfile* is the profile in which IBM FileNet Content Manager is deployed.

7. Add the JNDI factory and provider:

   ```
   -Djava.naming.factory.initial=%JNDI_CLIENT_FACTORY% ^
   -Djava.naming.provider.url=%JNDI_CLIENT_PROVIDER% ^
   ```

8. Point to the IBM FileNet Content Manager `WcmApiConfig.properties` file that you modified earlier in steps 2 and 3:

   ```
   -Dfilenet.wcmapiconfig="%VBR_HOME%/lib/WcmApiConfig.properties" ^
   ```

9. Add the CORBA configuration URL:

   ```
   -Dcom.ibm.CORBA.ConfigURL=%WAS_HOME%/profiles/myProfile/properties/s
   as.client.props ^
   ```

   where *myProfile* is the profile in which IBM FileNet Content Manager is deployed.

10. Save and close the RMI proxy connector file (`RMIBridge.bat` or `RMIBridge.sh`).

11. Set the connector properties using the Administration Tool. Insure that the connector property URLs are configured to match the values that are used in step 2:

    ```
    Remote Server URL: cemp:iiop://host:2809/FileNet/Engine
    Remote Server Upload URL: cemp:iiop://host:2809/FileNet/Engine
    Remote Server Download URL: cemp:iiop://host:2809/FileNet/Engine
    ```

Archived

### *IBM FileNet Content Manager Version 4.5 with connector hosted in an environment other than the IBM FileNet Content Manager server*

To configure the FileNet Content Manager connector for IBM FileNet Content Manager Version 4.5 running on IBM WebSphere Version 6.1.*x*, with the connector hosted in an environment *other than* the IBM FileNet Content Manager server, perform the following steps:

**Requirements:**

- You must have the IBM FileNet Content Manager Application Engine installed to obtain the libraries that are needed to configure the connector.
- If lockdown functionality is required, you must have IBM WebSphere Application Client Version 6.1.0.11 or later installed on the server that hosts the connector. FP42959 is used.

**Variables:** Refer to these variables, which are used in the following instructions:

- *AE_HOME*: The IBM FileNet Content Manager Application Engine installation directory, typically, *P8_HOME*/AE
- *ICI_HOME*: The Content Integrator installation directory where the RMI proxy connector for FileNet Content Manager is running
- *P8_HOME*: The IBM FileNet Content Manager installation directory
- *WAC_HOME*: The IBM WebSphere Application Client installation directory where the RMI proxy connector for FileNet Content Manager is running
- *WAS_HOME*: The IBM WebSphere Application Server installation directory running IBM FileNet Content Manager

1. Install the IBM WebSphere Application Client on the server that will host the FileNet Content Manager RMI proxy connector. The IBM WebSphere Application Client version must match the version of the IBM WebSphere Application Server that is installed on the IBM FileNet Content Manager system.

   During the installation of the client, enter the host name and the port for the IBM WebSphere Application Server that hosts IBM FileNet Content Manager.

2. Modify the `sas.client.props` file in the IBM WebSphere Application Server installation directory on the system that is running IBM FileNet Content Manager. Typically, this file is located in the *WAS_HOME*/profiles/*myProfile*/properties directory, where *myProfile* is the profile in which IBM FileNet Content Manager is deployed. Set the following

properties using the host name of the system that is running IBM FileNet Content Manager:

```
com.ibm.CORBA.sercurityServerHost=hostname
com.ibm.CORBA.loginSource=none
```

3. Copy the `sas.client.props` file that you modified in step 2 from the IBM FileNet Content Manager server to the server that will host the FileNet Content Manager RMI proxy connector. Place this file into the *WAC_HOME*/`properties` directory.
4. Copy the following files from the IBM FileNet Content Manager system to the server that will host the FileNet Content Manager RMI proxy connector. Place these files into the *ICI_HOME*/`lib` directory:
   - `Jace.jar`, `javaapi.jar`, and `WcmApiConfig.properties` files from the *AE_HOME*/`Workplace/WEB-INF/lib` directory
   - `UTCryptoKeyFile.properties` file, which is located in the *P8_HOME*/`FileNet/Authentication` directory
5. Edit the properties in the `WcmApiConfig.properties` file on the server that will host the FileNet Content Manager RMI proxy connector to refer to the IBM FileNet Content Manager server. Replace *host* with the host name or IP address where IBM FileNet Content Manager is running. Port 2809 is the application server default; modify it as necessary for your deployment:

   ```
   RemoteServerUrl=cemp:iiop://host:2809/FileNet/Engine
   RemoteServerUploadUrl=cemp:iiop://host:2809/FileNet/Engine
   RemoteServerDownloadUrl=cemp:iiop://host:2809/FileNet/Engine
   ```

6. In the same `WcmApiConfig.properties` file that you modified in step 5, add the following line to specify the key for locating a UserTransaction reference during an IBM FileNet Content Manager JNDI lookup:

   ```
   TxJndiKey=jta/usertransaction
   ```

Now that the connector environment has been configured, update the RMI proxy connector script file in the *ICI_HOME*/`bin` directory:

1. Add a *WAC_HOME* variable to the `RMIBridge.bat` or `RMIBridge.sh` file, setting it to the IBM WebSphere Application Client home directory:

   ```
   set WAC_HOME=C:/Progra~1/IBM/WebSphere/AppClient
   ```

2. Set the *JAVA_HOME* variable to point to the JVM of the IBM WebSphere Application Client installation. You might need to modify the PATH variable to reference the JVM:

   ```
   set JAVA_HOME=%WAC_HOME%/java
   set PATH=%JAVA_HOME%/bin;%PATH%
   ```

3. Optional: Add the two JAR files that you copied to the *ICI_HOME*/lib directory in step 4 (Jace.jar and javaapi.jar) to the *VBR_ALLJARS* variable:

   ```
   set VBR_ALLJARS=%VBR_ALLJARS%;VBR_HOME/lib/Jace.jar
   set VBR_ALLJARS=%VBR_ALLJARS%;VBR_HOME/lib/javaapi.jar
   ```

   Under most circumstances, these JAR files will be included in the Java classpath by the config.bat or config.sh file, because all JAR files are picked up from the *ICI_HOME*/lib directory, by default. However, if you have modified the config.bat or config.sh file for any reason for your environment, you must perform this step.

4. Add a *JNDI_CLIENT_PROVIDER* variable. Set this variable to the host name or IP address of the system running the IBM WebSphere Application Server for IBM FileNet Content Manager. Replace *host* with the host name or IP address where IBM FileNet Content Manager is running:

   ```
   set JNDI_CLIENT_PROVIDER=iiop://host:2809
   ```

5. Add a *JNDI_CLIENT_FACTORY* variable. Set this variable to the naming factory that is being used:

   ```
   set
   JNDI_CLIENT_FACTORY=com.ibm.websphere.naming.WsnInitialContextFactory
   ```

6. Set the following Java options to start the RMI proxy connector. These options are added after the **java** command and before the **RMIBridgeFactoryLauncher** is specified. Add Java options to include the directories in the client:

   ```
   -Djava.ext.dirs=%WAC_HOME%/plugins;%WAC_HOME%/java/jre/lib/ext;%WAC_
   HOME%/java/jre/lib;%WAC_HOME%/classes;%WAC_HOME%/lib;%WAC_HOME%/lib/
   ext ^
   ```

7. Add the JNDI factory and provider:

   ```
   -Djava.naming.factory.initial=%JNDI_CLIENT_FACTORY% ^
   -Djava.naming.provider.url=%JNDI_CLIENT_PROVIDER% ^
   ```

8. Point to the IBM FileNet Content Manager WcmApiConfig.properties file that you modified earlier in steps 5 and 6:

   ```
   -Dfilenet.wcmapiconfig="%VBR_HOME%/lib/WcmApiConfig.properties" ^
   ```

9. Add the CORBA configuration URL:

   ```
   -Dcom.ibm.CORBA.ConfigURL=%WAC_HOME%/properties/sas.client.props ^
   ```

10. Save and close the RMI proxy connector file (RMIBridge.bat or RMIBridge.sh).

Archived

11. Set the connector properties using the Administration Tool. Insure that the connector property URLs are configured to match the values that are used in step 5:

    ```
    Remote Server URL: cemp:iiop://host:2809/FileNet/Engine
    Remote Server Upload URL: cemp:iiop://host:2809/FileNet/Engine
    Remote Server Download URL: cemp:iiop://host:2809/FileNet/Engine
    ```

One last modification is required. You must configure the IBM WebSphere Application Server that is used by the IBM FileNet Content Manager system so that the WebSphere Application Client can communicate with it:

1. On the IBM FileNet Content Manager system, open the IBM WebSphere Application Server administrative console.
2. Select **Servers** → **Application Servers** → ***server1*** → **Ports** where *server1* is your application server name.
3. Change all instances of localhost to the IP address or the host name of the IBM FileNet Content Manager system. Leaving this setting as localhost prevents the RMI proxy connector from locating the IBM WebSphere Application Server running IBM FileNet Content Manager.
4. Restart IBM WebSphere Application Server.

### *IBM FileNet Content Manager Version 3.5.x*

To configure the FileNet Content Manager connector for IBM FileNet Content Manager Version 3.5.*x* running on Oracle WebLogic, perform the following steps (An RMI proxy connector is not required for this version, although you can use it if you want.):

1. Copy the `WcmApiConfig.properties` resource file from the IBM FileNet Content Manager server to the *`ICI_HOME`*`/bin` directory. Find the resource bundle in the `FileNet/Workplace/WEB-INF` directory of your IBM FileNet Content Manager installation location.
2. Edit the following properties in the `WcmApiConfig.properties` file that you copied to refer to the IBM FileNet Content Manager server:

    ```
    RemoteServerUrl
    RemoteServerUploadUrl
    RemoteServerDownloadUrl
    ```

3. Configure the classpath of the JVM hosting the FileNet Content Manager connector to include access to the IBM FileNet Content Java API JAR files:

    ```
    activation.jar
    javaapi.jar
    mailapi.jar
    ```

4. Set the connector properties using the Administration Tool.

## Connector properties

The FileNet Content Manager connector-specific properties that are presented in Figure 7-18 are configured using the Administration Tool.

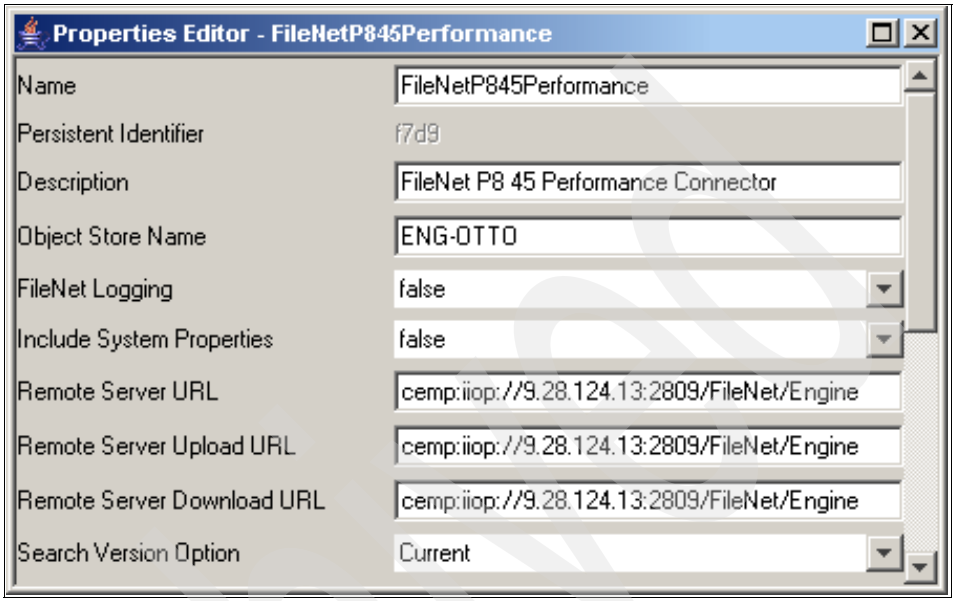



*Figure 7-18 Properties Editor for the FileNet Content Manager connector*

We list the properties and their definitions:

- **Object Store Name**

  This property is the IBM FileNet Content Manager object store that is used by the connector.

- **FileNet Logging**

  This property enables IBM FileNet Content Manager transaction logging for events that are initiated by the Content Integrator connector.

- **Include System Properties**

  This property indicates whether system properties are included with content retrieval using the Content Integrator connector. When this property is set to true, many system properties are retrieved in addition to the properties that are associated with the content's item class. Depending on your environment, this property can affect performance.

- **Remote Server URL**

  This property identifies the URL of the IBM FileNet Content Manager engine.

- **Remote Server Download URL**

  This property identifies the download URL of the IBM FileNet Content Manager engine.

- **Remote Server Upload URL**

  This property identifies the upload URL of the IBM FileNet Content Manager engine.

- **Search Version Option**

  This property determines whether a search request uses the current version or the released version of a document. The value for this property is either current or released. This property serves the same purpose as the Version Option property in the search section of the IBM FileNet Content Manager Workplace Application.

- **Use_Symbolic_Name** (custom property)

  When set to true, the IBM FileNet Content Manager symbolic name is used as the item class name. If the value is set to false, the IBM FileNet Content Manager display name is used as the item class name. The default setting is true.

- **USE_VERSIONSERIES_ID** (custom property)

  The USE_VERSIONSERIES_ID custom connector property controls which IBM FileNet Content Manager property is used as the identifier for content items. If this value is set to true, the version series ID is used as the identifier for content items. All versions of a document have the same version series ID. If this value is set to false, the document ID is used as the identifier for content items. Each version of a document has a separate document ID. The default setting is true.

  Folder items are not affected by this property, because folder items do not have version series IDs.

## Additional connector information

We include additional information about the Content Manager connector:

- **Searching the ID property**

  When searching with the ID property as the selection criteria, the value must be in the accepted IBM FileNet Content Manager global unique identifier (GUID) format. If an ID is specified in an invalid format, the search request will fail.

Archived

- **Full-text search**

  All properties in IBM FileNet Content Manager item classes are marked as not searchable with full-text searches; however, full-text searches of IBM FileNet Content Manager systems will return correct result sets.

  Full-text searches against Arabic characters will return unexpected results. This situation is an issue with the IBM FileNet Content Manager and not with Content Integrator.

- **User name property**

  The user name property must be set to the full user name that includes the IBM FileNet Content Manager domain name, for example, SysAdmin@FILENETV3. The user must exist in the IBM FileNet repository.

- **IBM FileNet Content Manager security**

  The FileNet Content Manager connector supports the advanced, enhanced, and the read-token security models in Content Integrator. You can use the token methods to allow read access to content through Content Integrator. With the advanced security model, application developers can set separate read, update, delete, and change security privileges for users and groups. The enhanced security model exposes security features of the content management system, such as the security schema, hierarchical groups, ACLs, and system privilege types.

- **Property names that include the ampersand (&) character**

  Using the ampersand (&) character causes errors if it is included in the property names of content items. These errors can occur when you attempt to create a content item or when you try to retrieve a content item.

- **Long names security**

  You must use long names when you query or update security in the IBM FileNet Content Manager connector. If security is updated with a short name, the following error occurs:

  ```
  class com.venetica.vbr.client.PermissionDeniedException
  COEN2201E:  The group or user does not have a registered security
  account with FileNet.
  ```

- **Locking and unlocking content**

  You can configure the FileNet Content Manager connector to enable Content Integrator to lock and unlock content that is stored in IBM FileNet Content Manager.

  The lockDownRecord() operation modifies the content's security to limit access privileges for all users (except for the user that is currently logged onto the content management system) and transfers the content item's owner to the user identifier of the user that is currently logged on. The

Archived

lockDownRecord() operation also grants administrative-level permissions to the content's owner.

See the Content Integrator Information Center for more information about locking and unlocking content with IBM FileNet Content Manager.

## 7.3.6 Microsoft Windows SharePoint connector configuration

The Microsoft Windows SharePoint connector enables Content Integrator access to content that is stored in a Microsoft Windows SharePoint Services content management system. The Microsoft Windows SharePoint connector uses the Microsoft SharePoint Services .NET API. We provide configuration details for Microsoft Windows SharePoint 2007/SharePoint Services 3.0 and Microsoft Windows SharePoint 2003/SharePoint Services 2.0.

### Connector environment setup

There are two components to the Microsoft Windows SharePoint connector: a Java-based component and a .NET-based application. The setup and configuration of the Microsoft Windows SharePoint connector requires that the .NET portion of the connector is installed on the Microsoft Windows SharePoint Services server. Perform this step first. Next, perform the actual connector configuration.

To install and configure the Microsoft Windows SharePoint .NET application:

1. Determine which .NET application to use:
   - If connecting to Microsoft Windows SharePoint 2007/SharePoint Services 3.0, use the files that are located in the Content Integrator installation directory: `ICI_HOME/bridges/sharepoint2007/IICESP2007WebService`.
   - If connecting to Microsoft Windows SharePoint 2003/SharePoint Services 2.0, use the files that located in the Content Integrator installation directory: `ICI_HOME/bridges/sharepoint/IICESPWebService`.
2. After determining which .NET application to use in step 1, copy the contents of the directory to an appropriate folder on the server where the Microsoft Windows SharePoint Services is running. For example, copy the files to the `C:/SharePointWebServices` directory.
3. On the server where Microsoft Windows SharePoint Services runs, start the Internet Information Services (IIS) Manager.
4. Create a new Virtual Directory on the same Web on which SharePoint runs. Name the new virtual directory `SharePointWebServices`.
5. Right-click the new virtual directory that you created in step 4, and click **Properties**.

6. On the Virtual Directory properties tab, set the properties in Table 7-6.

*Table 7-6 Virtual Directory properties tab*

| Property | Value/Action |
|---|---|
| A directory located on this computer | Selected. |
| Path or Local path | Specify the directory that contains the .NET application files that you copied in step 2. |
| Read | Selected. |
| Application name | Select **Create** to create the application. The default application name will be created with the value that you entered in step 4 for the virtual directory name. |
| Execute permissions | Scripts only. |

7. On the Directory Security tab:
   – Choose to edit the Authentication and access control option.
   – Select **Basic authentication** (password is sent in clear text).
   – If using a domain, enter a valid value for the Default domain, which is the domain that users will be authenticated against when attempting to access the server.
   – Click **OK**.
8. If you are connecting to Microsoft Windows SharePoint 2007/SharePoint Services 3.0 system, skip to step 15, “Verify the .NET application.”
9. In the Internet Information Services (IIS) Manager, right-click to select **Default Web Site** → **All Tasks** → **SharePoint Central Administration**. The Central Administration page opens in your Web browser.
10. Select **Configure Virtual Server Settings**. The Virtual Server List page opens in your browser.
11. Select the virtual server where Microsoft Windows SharePoint Services is installed. The Virtual Server Settings page opens in your Web browser.
12. Select **Define Managed Paths** under Virtual Server Management.
13. In the Add a New Path section, enter the name of the virtual directory for the Microsoft Windows SharePoint Services connector .NET application that you created in step 4: SharePointWebServices.
14. Select **Excluded path** as the path type, and click **OK**.

Archived

15. Verify the .NET application.

    You can verify the .NET application installation and configuration by opening the following URL on the server where the Microsoft Windows SharePoint Services is running:

    `http://localhost/SharePointWebServices/service.asmx`

    A list of Web services is displayed. You have completed the .NET application installation and configuration.

16. Set the connector properties using the Administration Tool.

**SSL:** To use SSL with the Microsoft Windows SharePoint connector, see "Configuring the connector to use Secure Sockets Layer" on page 378.

## Connector properties

The Microsoft SharePoint connector-specific properties presented in Figure 7-19 are configured using the Administration Tool.

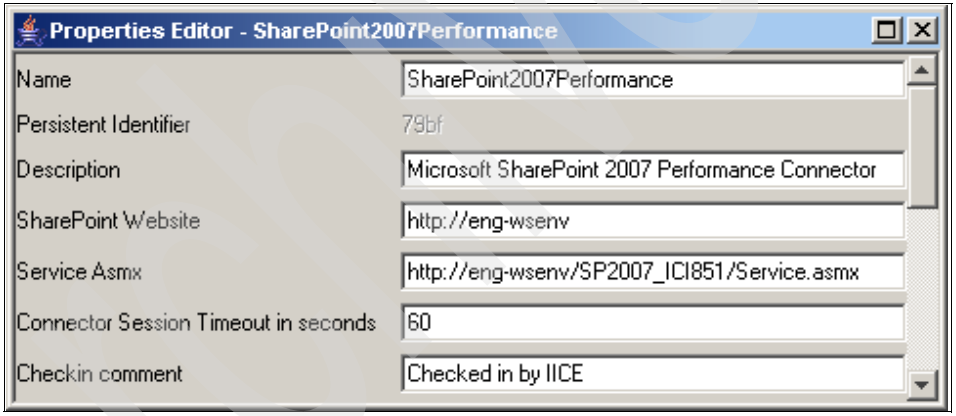



*Figure 7-19 Properties Editor for the Microsoft SharePoint connector*

We list the properties and their definitions:

- **SharePoint Website**

  This property is the URL of the Microsoft Windows SharePoint Services Web site. If your environment includes more than one subsite, configure a separate Content Integrator connector for each subsite.

- **Service Asmx**

  This property is the URL of the Content Integrator Microsoft SharePoint connector .NET application running on the Microsoft Windows SharePoint Services system, for example:

  `http://sharepointserver/SharePointWebServices/service.asmx`

- **Connector Session Timeout in seconds**

  This property is the number of seconds that Content Integrator waits during an operation before it closes the connector session. There is no default value for this property; however, the Web services default of 60 seconds will be used if there is no specified value for this property.

- **Checkin comment**

  The checkin comment property is used to associate a comment with content that is checked into the content management system. The checkin comment is associated with newly checked-in version of the content.

- **MimeTypes** (custom property)

  The Content Integrator Microsoft SharePoint connector uses an internal mapping of native content file extensions to determine MIME types. The mapping is based on the content filename extension.

  A default set of MIME type mappings is provided with the connector configuration; however, you can easily make modifications or additions. Follow this format for these custom properties:

  Property Name: `mimeTypeMapping`*xxx* where *xxx* is a unique number
  Property Value: `extension:MIME type` to use

  For example, to associate the MIME type mapping image/tiff with the `.tix` filename extension, use the property value:

  `tif:image/tiff`

- **NonQueryableProps** (custom property)

  The NonQueryableProps custom connector property for the Microsoft SharePoint connector is a pre-populated list of comma-delimited names of item properties that must *not* be included in the search criteria of Content Integrator searches. Attempting to use these properties as search criteria results in either a Microsoft Windows SharePoint Services exception or unexpected results.

  The default list is provided with the creation of the connector configuration:

  `Author, Created_x0020_Date, Edit, Editor, EditUser, FileSizeDisplay, IsCurrent, Last_x0020_Modified, LinkCheckedOutTitle, LinkFilename, LinkFilenameNoMenu, LinkObjective, NameOrTitle, LinkObjectiveNoMenu,`

Archived

`Reply, RequiredField, SelectedFlag, _CheckinComment, ParentVersionString, ParentLeafName`

Use the Content Integrator Administration Tool to view and edit the list of item property names.

## Additional connector information

We list additional information about the Microsoft SharePoint connector:

- **Searching with the Microsoft SharePoint connector**

  Certain limitations exist when searching with the Microsoft Windows SharePoint connector:

  - Item classes are required if specifying search properties or search criteria in a search request. A search container is not required, but it is used if included in the search request.
  - Full-text searches do not allow item class or search container. Full-text searches are recursive.
  - You cannot include the item properties that are listed by name in the NonQueryableProps custom property as search criteria. For more information, see “NonQueryableProps (custom property)” on page 375.
  - Content Integrator operations, such as searching and retrieving, use Microsoft Windows SharePoint Services InternalName values.
  - You can search for a list item using item finders. You cannot search for attachments of list items.
  - You cannot combine property-based searches with full-text searches.
  - When properties of type DateTime are used in property searches, only the date portion of the property value is used, the time information is not used in the search.

- **SharePoint Services group requirements**

  The Microsoft Windows SharePoint Services connector has the following group requirements:

  - Users that belong only to the SharePoint Guest group cannot access content using the connector. Insure that users requiring access to content through the connector belong to more than only the Guest group.
  - Microsoft Windows SharePoint Services users without the Browse Directories user privilege cannot see folder hierarchies in document libraries. Instead, all content is presented at the top level of the document library. To view folder hierarchies, add the Browse Directories privilege to the group to which the user belongs. By default, the Reader group does

Archived

not have the Browse Directories privilege. The Contributor, Web Designer, and Administrator groups have the Browse Directories privilege.

- **Item class support**

  Each Microsoft Windows SharePoint Services document library is considered an item class that is assigned to each content item in the document library. Document libraries are treated as top-level folders in Content Integrator.

  Each Microsoft Windows SharePoint Services list is considered an item class that is assigned to each content item in the list. Lists are treated as top-level folders in Content Integrator.

  The Microsoft Windows SharePoint Services connector supports the Content Integrator actions for the item class types that are presented in Table 7-7.

*Table 7-7 Microsoft Windows SharePoint connector item class type support*

| Content Integrator function | Document libraries | Lists | Discussion boards |
|---|---|---|---|
| Read and search | Yes | Yes | Yes |
| Browse | Yes | Yes | Yes |
| Create, update, and delete | Yes | No | No |

  Functionality that is indicated as unsupported can result in an exception returned from Microsoft Windows SharePoint Services if the functionality is attempted.

- **Modifying properties and native content**

  To reflect Microsoft Windows SharePoint Services system behavior, you can modify native content if the content item is checked out; however, you cannot modify the content properties. For content that is not checked out, you can modify both the native content and the content properties.

- **Attachments**

  Microsoft Windows SharePoint Services list items can include multiple attachments. Content Integrator treats each attachment as a separate page of a multiple page content item. SharePoint attachments cannot be versioned.

Archived

- **MIME types and versions**

  The Microsoft Windows SharePoint Services system does not store the MIME-type information of previous versions of content. The MIME types of all versions of a document are assumed to be the same as the MIME type of the current version.

- **Streaming support**

  The Microsoft Windows SharePoint connector supports native content retrieval and upload using streams, as long as the connector and the repository run on the same computer.

## Configuring the connector to use Secure Sockets Layer

The Content Integrator Microsoft Windows SharePoint connector contains a key .NET component that is deployed on the same server on which Microsoft Windows SharePoint Services is running. If the Internet Information Services (IIS) hosting SharePoint Services has Secure Sockets Layer (SSL) enabled, the Content Integrator Microsoft Windows SharePoint connector also needs to have SSL enabled.

To configure the Microsoft Windows SharePoint connector to use SSL, perform the following steps:

1. Enable the server to use SSL. By default, the IIS application server runs through HTTP access. You must create a certificate request for the server. The server is the same system where IIS is running the Microsoft Windows SharePoint Services instance:
   - Start the IIS Manager tool.
   - Right-click the Web site, and select **Properties**.
   - Select the **Directory Security** tab.
   - Go to the Secure communications frame, and click **Server Certificate**.
   - Select **Prepare the request now, but send it later**.
   - Enter a file name for the certificate.
   - Add the common name setting. Use the host name of the system.
   - Fill in the rest of the information accordingly. Then, select **Finish**.
2. After receiving the requested certificate from your certificate authority, install the certificate to your Web server:
   - Copy and paste the certificate text from the certificate authority's e-mail that you received into a new text document. Name the document:

     `certificate.cer`
   - Start the IIS Manager tool.

- Right-click the Web site, and select **Properties**.
- Select the **Directory Security** tab.
- Click **Server Certificate**.
- Select **Process the pending request and install the certificate**.
- When prompted for the certificate file, give the file path to the certificate file that you saved in step 2.
- Accept the defaults for the rest of settings. Note the assigned port number.

3. Enable SSL in the Web server:
   - Start the IIS Manager tool.
   - Right-click the Web site, and select **Properties**.
   - Select the **Directory Security** tab.
   - Select to edit Secure communications.
   - Select **Require secure channel (SSL)**.
   - Restart the Web site. To test the configuration, follow the directions that are provided in the certificate authority's e-mail as needed for your Web browser.
4. Configure the Microsoft Windows SharePoint connector to use SSL. In the previous step, the client configuration tested was the Web browser. For this step, the client is the RMI proxy connector that consumes Web services:
   - Configure your JVM environment to use the certificate that is generated by the certificate authority. Locate the key tool in the *JAVA_HOME*/bin directory.
   - Verify that the certificate that you have is valid. In the tool, enter this command:

     ```
     keytool -printcert -file $CERT_HOME/certificate.cer
     ```

     *$CERT_HOME* is a directory where you have made a copy of your certificate.cer file from step 2.
   - Import the certificate from the certificate authority to your JVM environment using this command:

     ```
     keytool -import [-keystore keystore] -file
     $CERT_HOME/certificate.cer
     ```

     The -keystore option tells the key tool to use a particular key store. If none is specified, the tool uses the default key store. On AIX, Hewlett-Packard UNIX (HP-UX), Linux, and Solaris, the saved file is located in the *user.home* variable.

Archived

– To verify that the certificate was successfully imported, enter this command:

```
keytool -list
```

– Add the following text to the batch file that starts your RMI proxy connector for Microsoft Windows SharePoint:

```
$JAVA_HOME/bin/java -Djavax.net.ssl.trustStore=$HOME/.keystore
```

where *$HOME* is the directory containing `.keystore`.

– Restart your RMI proxy connector for Microsoft Windows SharePoint.

5. Configure your Microsoft Windows SharePoint connector to use HTTPS using the Content Integrator Administration Tool. Set the SharePoint Web site and Service Asmx connector properties appropriately.

## 7.3.7 Open Text Livelink connector configuration

The Open Text Livelink connector enables Content Integrator access to content that is stored in an Open Text Livelink content management system. The Livelink connector uses the Open Text Livelink Java API. We provide configuration details for Open Text Livelink Version 9.7.1, Version 9.7, and Version 9.6.0.*x*.

### Connector environment setup

The Livelink connector is one of the simplest Content Integrator connectors to configure. The only requirement is that the connector JVM has access to the Open Text Livelink Java API JAR file.

To configure the Livelink connector:

1. Configure the classpath of the Java virtual machine hosting the Livelink connector to include the Open Text Livelink Java API JAR file named `lapi.jar`. You can find the `lapi.jar` file in the `/lib` directory of your Open Text Livelink system installation.
2. Set the connector properties using the Administration Tool.

### Connector properties

The Livelink connector-specific properties that are presented in Figure 7-20 on page 381 are configured using the Administration Tool.

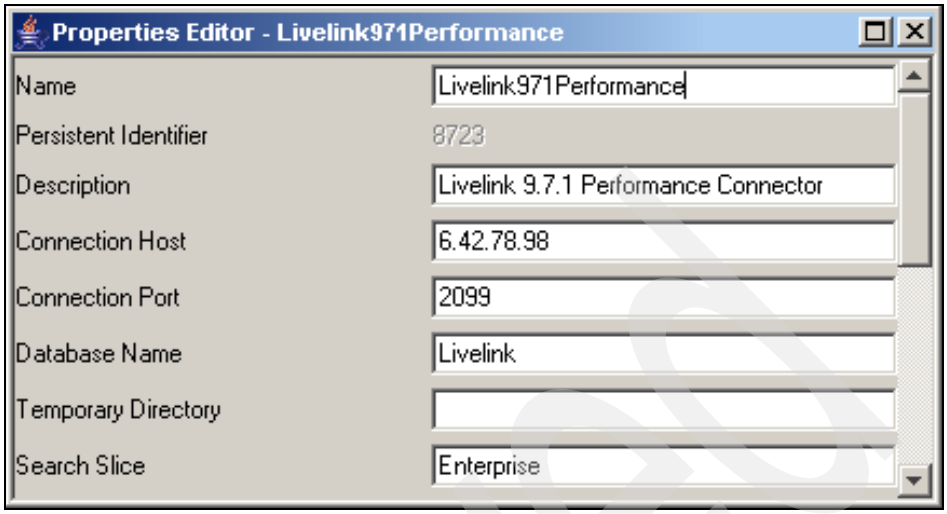



*Figure 7-20 Properties Editor for the Livelink connector*

We list the properties and their definitions:

- **Connection Host**

  This property is the name of the server where the Open Text Livelink content management system is running. For a secure low-level application programming interface (LAPI) connection, the host name must be the name of the Web server. You can also use an IP address instead of the name.

- **Connection Port**

  This property is the port number where the Open Text Livelink server receives HTTP requests. In a standard Open Text Livelink installation, the plain text port number is 2099. The secure port is 443. Content Integrator defaults to 2099, which is the plain text port.

- **Database Name**

  This property is the name of the Open Text Livelink connection that is used to connect to the database. This name is the same Database Name value that is entered during the database creation step of the Open Text Livelink system install.

  If this property is left blank, the system uses the default connection that is assigned to the *`dftConnection`* variable in the general section of the `opentext.ini` file. You can find the `opentext.ini` file in the `/config` directory of your Open Text Livelink system installation.

Archived

- **Temporary Directory**

  This property is a temporary location where the Livelink connector places content that is loaded into or extracted from the Open Text Livelink content management system. If specified, the directory must be available and accessible on the server where the Livelink connector is deployed. To use the default temporary location of *ICI_HOME*/temp, leave this property blank.

- **Search Slice**

  This property is the name of the Open Text Livelink search broker or slice to use for running searches. Search brokers are created in the Open Text Livelink content management system. The default value is Enterprise.

- **LockDownPrincipal** (custom property)

  This property is the user name or group name to use as the document owner after a document is locked. For more information about the use of this property, see "Locking and unlocking content" on page 382.

- **LockDownPrincipalType** (custom property)

  This property is the lockdown principal type. If the LockDownPrincipal custom property is set, this value must be either user or group. If the LockDownPrincipal custom property is not set, this value can be blank.

## Additional connector information

We provide additional information about the Livelink connector:

- **Lexical ordering**

  In Content Integrator, lexical sorting is case sensitive. Open Text Livelink sorting is case *insensitive*. Therefore, the order of the results that are returned using Content Integrator can differ from those results that are returned by native Open Text Livelink tools. If search results are sorted and a maximum number of results are specified, the actual result collection can differ from what is expected. One work-around is to not use a maximum results value when a sort specification or an order-by property is provided. Another option is to sort the results collection after it is returned from a search request.

- **Livelink security**

  The Livelink connector supports the advanced, enhanced, and the read-token security models in Content Integrator. The connector can retrieve and apply the security schema, user names, groups, and ACLs from the Open Text Livelink content management system.

- **Locking and unlocking content**

  You can configure the Livelink connector to enable Content Integrator to lock and unlock content that is stored in Open Text Livelink. Using the Administration Tool, define the LockDownPrincipal custom property. The

LockDownPrincipal property is the name of the user or group that becomes the content owner after content items are locked.

The lockDownRecord() operation modifies the security of content so that users and groups in the ACL have no higher than the Version privilege, and it grants the Delete and Change Security privileges to the content item's owner. The operation also changes the content item's owner to the user or group that is specified in the LockDownPrincipal custom property.

The value that is specified in the LockDownPrincipal custom property must be the name of the user that will call the lockDownRecord() operation or the name of a group of which the user is a member. The user name or group name must be valid in the Open Text Livelink system. Additionally, the LockDownPrincipalType must be set to either user or group.

Locking and unlocking content is enabled at the content object level only.

- **MIME type for BMP files**

  Open Text Livelink does not include the MIME type for BMP files. Edit the Open Text Livelink `mime.types` configuration file by adding the following line:

  ```
  image/bmp   bmp
  ```

  You can find the `mime.types` configuration file in the `/config` directory of your Open Text Livelink system installation. You can also specify any other MIME types that are not already present in the configuration file. You must restart the Open Text Livelink server before configuration changes take effect.

- **Null values and the order-by property**

  You can specify that results that are returned from a search request are sorted using the Query.setOrderBy(String) operation. However, Open Text Livelink content containing null values for the specified order-by property is not included in the results. We recommend that you do not use order-by properties if there is the potential to have null values for the specified property.

Archived

Archived

8

# Understanding the client APIs

This chapter describes the programming interfaces of IBM Content Integrator that you can use to access content from disparate repositories. We introduce the basics of the Content Integrator Java API and the two sets of Web Services APIs, and we cover advanced topics in the Java API. To help you get started, we walk through a sample client application and share many hints and tips along the way.

We cover the following topics in this chapter:

- Introduction to the Java API
- A sample client application
- Advanced Java API topics:
  - Session pools
  - Single sign-on
  - Configuration API
  - Federated Query Transformation
  - Security models
  - Complex properties
- Web Services

© Copyright IBM Corp. 2010. All rights reserved.

## 8.1 Introduction to the Java API

The Content Integrator Java interface consists of two distinct parts: the *Client Application Programming Interface (API)* and the *Connector Service Provider Interface (SPI)*. The Java Client API consists of classes and methods that allow a client application to communicate with repositories through Content Integrator. In contrast, the Connector SPI consists of classes and methods that represent a standard interface between the core of Content Integrator and the connectors. Figure 8-1 depicts the two programming interfaces.

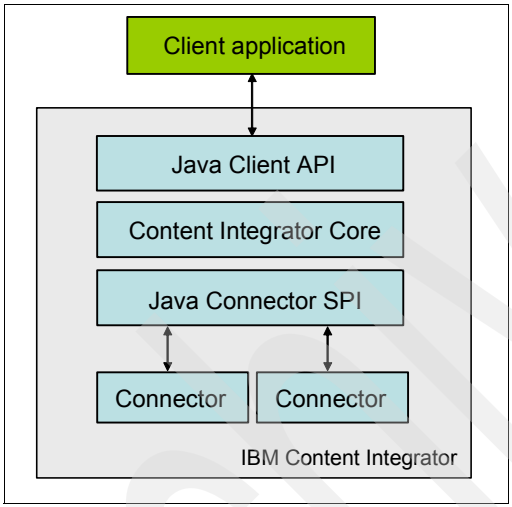



*Figure 8-1 The Content Integrator Java API and SPI*

The *Java Client API* is a rich, object-oriented content management API. It allows developers to write content management and content integration applications generically, through a single, easily mastered set of Java classes. It supports federated and single-repository search, as well as the retrieval, creation, update, and deletion of content in any back-end repository. The API represents a super-set of content management capabilities and is not tied to any specific repository. To reflect the purpose of the API, it is sometimes also referred to as the *Integrate API*.

The Java Client API also supports functionality beyond content integration and federation. For example, it allows you to access and modify configuration parameters of connectors, data maps, and session pools. This component of the Java API is referred to as the *Configuration API*.

Not mentioned in Figure 8-1, there are additional smaller SPI extension points in the product. These extension points allow you to customize Content Integrator

according to your requirements. For example, you can plug in a custom logger, an encryption framework, a federated query pre-processor and post-processor, a Single Sign-On (SSO) credential store, an image converter, and more.

### 8.1.1 Setting up the environment

Content Integrator 8.5.x requires the Java 1.5 Runtime Environment and supports most of the major operating systems and hardware platforms. You can find the complete list of system requirements at this Web site:

http://www.ibm.com/support/docview.wss?rs=86&uid=swg27015930

Enabling a client application for the Content Integrator Java API requires two simple setup steps:

- Add the `vbr.jar` library file to the application's classpath. This library contains all of the classes of the Java API and core components of Content Integrator. You can find the `vbr.jar` library file in the *`ICI_HOME`*`/lib` directory.
- Add the Java virtual machine (JVM) property vbr.home to the application's JVM. Specify the home directory of Content Integrator as the property value. This property is required to look up configuration files, such as the `config.xml` file or the `vbr_services.properties` file.

Assuming Content Integrator is installed in the `/opt/ICI` directory, the following UNIX command starts the sample application MyClientApp with the correct classpath and JVM property:

```
java -classpath /opt/ICI/lib/vbr.jar:. -Dvbr.home=/opt/ICI MyClientApp
```

For both the classpath and the vbr.home property, enclose any spaces in the path with double quotation marks.

**Third-party libraries:** Content Integrator bundles third-party libraries inside the `vbr.jar` library file. For example, the `vbr.jar` library file contains the class files of Apache log4j, Apache Axis2, and JIntegra. If your application happens to use the same third-party library as Content Integrator, you might encounter classpath errors if the library versions conflict. You can resolve these errors by either switching the order of the libraries in the classpath or by using a custom class loader in the part of the application that interacts with Content Integrator.

#### Local and remote connectors

The environment setup that we just described assumes that the connector is running in a separate JVM in a Remote Method Invocation (RMI) proxy connector server. Content Integrator looks up the location of the RMI proxy connector server in the connector configuration and forwards the client request to the

connector over RMI. The RMI proxy connector server is the recommended deployment choice for connectors, because it offers load balancing and fault tolerance (6.5.3, “RMI proxy connector server” on page 309).

Because of this assumption, the classpath that we have shown does not contain the connector library itself. However, it is possible to host the connector in the same JVM as the client application. In this case, there are additional setup steps that are required. You need to add the connector jar file and any third-party libraries, which are required to connect to the repository, to the client classpath. The connector jar files are located in the *ICI_HOME*/ejb directory. The setup steps differ from connector to connector, so read the connector configuration instructions in this book or in the information center and apply the same settings to your client environment.

## 8.1.2 Javadocs and samples

The Java Client API is well documented with Javadocs and samples.

### Javadocs

An HTML representation of the Javadocs is located in the docs/integrate directory in the installation directory of Content Integrator. There are separate directories for API and SPI:

*ICI_HOME*/docs/integrate/api/index.html
*ICI_HOME*/docs/integrate/spi/index.html

The HTML Javadocs are a good way to get an overview of the available classes and methods. We recommend to start with the classes User, Repository, Query, and RepoItem.

### Samples

Another excellent way to get familiar with the Client API is to use the Java commandline folder samples:

*ICI_HOME*/docs/examples/java/commandline
*ICI_HOME*/docs/examples/java/swing

The commandline folder contains the bulk of the sample programs. Each sample demonstrates a particular operation, such as retrieving a new content item, updating an item, or performing a query. We recommend starting with the RepoTest, ExecQuery, and GetRepoItem examples. We encourage you to look at the sample source code, because it shows all of the API methods that are related to an operation and contains many useful comments.

The swing folder contains the RepoBrowser sample. This sample is a more comprehensive client that offers a Swing-based graphical user interface to all configured repositories. It allows you to browse the repository's folder structure, look at the metadata and native content of documents, and perform queries. The query function is accessible through right-clicking the connector name, and it allows you to search one repository at a time.

All of the samples are precompiled and can be executed immediately after installation. To run a sample, use the `run_samples.bat` or `run_samples.sh` file in the *`ICI_HOME`*`/bin` directory:

```
cd C:\Program Files\IBM\ContentIntegrator\bin
run_sample commandline.RepoTest MyConnector username password
run_sample swing.RepoBrowser
```

On Windows, you can also run the samples using a shortcut by selecting **Start** → **All Programs** → **Content Integrator** → **Commandline Samples**. This shortcut brings up a command prompt in the correct folder and lists all available samples.

**Tips for running the command line samples:**

- The samples require input parameters, such as connector name, user name, password, and item ID. To discover which parameters apply to a particular sample, run the sample one time without any parameters.
- Enclose parameter values that contain spaces in double quotation marks.
- For query samples, enclose selection criteria literals in single quotation marks, even if they are numeric, for example:

  `selectionCriteria="lastName LIKE 'S*' AND age>'45'".`
- Parameter names are case sensitive, so follow the expected format.

Feel free to copy the sample code and reuse it in your client application. However, take the time to look at the code and understand what you are copying. Verify that the code matches your needs.

### 8.1.3 Major Java API classes

A client application mostly interacts with classes from the package com.venetica.vbr.client[1]. Table 8-1 on page 390 lists the primary classes in that package.

[1] This package name stems from Content Integrator's former product name "VeniceBridge" and the former company name "Venetica".

*Table 8-1 Major Java client API classes*

| Class name | Purpose |
|---|---|
| User | A connection to Content Integrator. This class is used to obtain Repository instances. |
| Repository | A single user's connection to a repository. This class supports logon, logoff, creating and retrieving repo items, retrieving item classes, and other functions. |
| RepoItem | The common base class for repository items, such as folders and content. This class supports item naming and identification, permissioning, and reading and writing of properties. |
| Content<br>extends RepoItem | A piece of content in a repository. This class has all functionality of RepoItem, plus support for versioning, checkin/checkout, and native content. |
| Folder<br>extends RepoItem | A repository item capable of containing other content or folders, thus providing a navigable hierarchical view of a repository. |
| ItemClass | A category for repository items with similar characteristics. Repositories generally support multiple item classes, each with potentially separate properties. |
| Property | A name/value pair that is associated with an item in the repository. The entirety of the properties of an item represents the item's metadata. |
| PropertyDescription | Metadata about a particular property. Includes the property's data type, minimum/maximum values, legal values, and more. |
| Query | Search class for items in a single repository. Supports metadata and full text search. |
| MultiQuery | Search class for items across multiple repositories simultaneously, which is also known as federated search. Provides a single combined result set. |
| ItemFinder | Retrieval utility that returns all items in one or more folders. This class is most suitable for browsing containers. It can also be used as a lightweight metadata retrieval tool. |

Archived

| Class name | Purpose |
| --- | --- |
| RepositoryProfile | Detailed information about a repository's capabilities. The Java API exposes a super-set of the capabilities of most repositories, but not every repository supports all of them. Clients can use this class to check for a certain capability before attempting to invoke that functionality. |

The bulk of your client application code can be written with these classes. In 8.2, "A sample client application" on page 391, we take a close look at a sample client application and explain each of these classes in detail.

**Using Java interfaces:** Many Java API classes implement Java interfaces. For example, the ResultSet and SimpleResultSet classes implement the IResultSet interface; the Content and Folder classes implement the IContent interface; and the Query and MultiQuery classes implement the IQuery interface. Use interfaces as the type of variables and parameters instead of concrete classes to make your client application more generic and flexible to changes in the future.

## 8.2 A sample client application

In this section, we examine a sample client application in detail. We explain the following Java API topics:

- Basic and secure login
- Federated and single-repository queries
- ResultSet operations
- Retrieval of metadata and native content
- ItemFinder
- Locale and time zone support

You can read about additional topics in 8.3, "Advanced Java API topics" on page 412.

### 8.2.1 Basic and secure login

To obtain a repository connection, the client first needs to initialize a User object. The User object can be viewed as an connection handle by which Content Integrator uniquely identifies a client application. Next, the client application obtains an instance of the Repository class from the User object. A repository

instance is always tied to a particular connector. Therefore, the client needs to specify the connector name in the method getRepositoryByID(), as configured in the Administration Tool. Finally, the client logs in to the repository by providing a user name and password. The third parameter of the logon method is optional and usually left blank. If supported by the connector, this parameter can be used to pass authentication-related data to the connector, such as a Windows NT® domain name or a security certificate. Example 8-1 shows the basic login sequence.

*Example 8-1 Basic login sequence*

```
// Connect to Content Integrator
User repoUser = new User();
repoUser.initialize();

// Retrieve the specified repository
String connectorName = “P8_45”;
Repository repo = repoUser.getRepositoryByID(connectorName);

// Log into the repository
repo.logon(user, password, "");
```

Alternatively, you can use a *secure login*. With a secure login, the client program places the authentication information into an AuthBundle and seals it using an encryption algorithm. Example 8-2 shows the steps of a secure repository login.

*Example 8-2 Secure login sequence*

```
// Connect to Content Integrator
User repoUser = new User();
repoUser.initialize();

// Retrieve the specified repository
String connectorName = “P8_45”;
Repository repo = repoUser.getRepositoryByID(connectorName);

// Create an AuthBundle
AuthBundle bundle = new AuthBundle(user, password);

// Instantiate a BlowfishSealer
BlowfishSealer bS = new BlowfishSealer();

try {
   // Encrypt the AuthBundle
   bS.seal(bundle);
```

```
} catch (EncryptionException e) {
   e.printStackTrace();
   return;
}

// Log into the repository.
repo.encryptedLogon(bundle);
```

The AuthBundle contains the encrypted user name and password. Depending on the location of the connector, the transfer of the bundle to the connector can involve a network trip to the RMI proxy connector server. The connector unseals the AuthBundle and logs into the repository. Certain connectors allow a secure login to the repository, as well, for example, through Secure Sockets Layer (SSL). Check the connector documentation in the information center for more details.

The default encryption implementation that is provided by Content Integrator is based on the symmetric Blowfish algorithm. Symmetric encryption requires the same secret key on the sender side and on the receiver side. Therefore, a secure login call only succeeds if the exact same key file is available to the client application and the connector. On both sides, the key file must be placed in the directory specified by vbr.home. The default key file generated by the Content Integrator installer is `BlowfishKey.ser`. You can generate a new key file with the class com.venetica.vbr.crypto.BlowfishKeyGenerator. You can even plug in a custom encryption algorithm. For more details, refer to the security section in the Information Center.

At the end of a client program, each repository connection must be closed and the User object must be released. Example 8-3 shows a logoff sequence. The method calls apply to both basic and secure login.

*Example 8-3 Logoff sequence*

```
if (repo != null && repo.getLogonID() != null) {
   try {
       // Logoff from repository
       repo.logoff();
   } catch (Exception ex) {
       ex.printStackTrace();
    }
} if (repoUser != null) {
   try {
       //Free resources
       repoUser.freeInstance();
    } catch (Exception ex) {
       ex.printStackTrace();
```

```
    }
}
```

**Session pooling and SSO:** Content Integrator offers optional components for session pooling and Single Sign-On (SSO). If you want to use either of these components, the sequence of login and logoff calls differs from the code that is shown in this section. Refer to 8.3.1, “Session pools” on page 412 and 8.3.2, “Single sign-on” on page 421 for code samples.

### Looking up configuration data

You configure connectors, data maps, session pools, and other components using the Content Integrator Administration Tool. These settings are persisted to the `config.xml` file. By default, this file is located in the home directory of Content Integrator. Optionally, a configuration server can be started to enable remote access to the configuration file. Client applications can use any of the following options to point Content Integrator to the configuration location:

- Set the JVM property vbr.as.direct.rmi.urls to point to one or more configuration locations. This property is a comma-separated list of URLs of configuration servers or configuration files, for example: `rmi://server1:1250/ConfigServer,rmi://backup:1251/ConfigServer` or `C:\configs\config.xml,Y:\backup\config.xml`. The implementation loops through the entries in the list and uses the first location that works.
- Set the JVM property vbr.home to point to the installation directory of Content Integrator. This directory is the default location of the `config.xml` file.
- Use the Configuration API and pass an in-memory representation of the configuration to Content Integrator. See Example 8-26 on page 427.

Content Integrator loads the configuration during the call User.initialize().

There is a similar JVM property titled vbr.configuration.server.rmiurl. This property also refers to the configuration server. However, it is only used by the configuration server launcher program to assign a URL to the newly started server. It is not used by the client application to refer to a configuration location.

## 8.2.2 Federated and single-repository queries

Depending on whether you want to search one or more repositories at a time, you need to use separate Java API classes. The Query class is restricted to one repository only, while the MultiQuery class can be used to search one or more repositories. Both classes are equal in terms of supported search options.

Example 8-4 shows how to create the two query objects. You obtain a Query instance through the Repository object. In contrast, a MultiQuery instance is obtained through the User object. After creating the MultiQuery instance, you need to add each repository that must be searched. A repository must be logged on prior to adding it to MultiQuery.

*Example 8-4 Creating a Query and a MultiQuery*

```
//Get a query for this repository
Query query = repo.createQuery();

// Get a multiquery for two repositories
MultiQuery fedQuery = repoUser.getMultiQueryInstance();
fedQuery.addRepository(repo1);
fedQuery.addRepository(repo2);
```

**Note:** Content Integrator uses the terms *selection criteria* and *selection properties* for specific parts of a query request. Selection criteria refers to search conditions on property values, such as `customerID='011'`. Selection properties refers to those properties that must be returned as columns in the result set, for example, customerID or customerName. You can map these two terms to an SQL query:

SELECT selection properties
FROM search container or item class
WHERE selection criteria
ORDER BY selectionProperty1 ASC, selectionProperty2 DESC

Property names that are used in query selection criteria or selection properties can be either the property names, as defined in the repository, or data map element names. You typically use data map element names in federated queries. They enable the client to construct a query with generic property names despite the fact that the properties in each repository have separate names. To use data map element names, you must associate a data map with the query object. Use the method setDataMapInUseName() and specify the data map name, as defined in the Administration Tool. See Example 8-5.

*Example 8-5 Associating a data map with the MultiQuery object*

```
//associate a data map with the MultiQuery object
fedQuery.setDataMapInUseName(dataMapName)
```

You can use a data map in a single-repository query, as well. You can use a data map if the repository property names are long or cryptic and you want to present the user with more readable property names.

Property name mapping through data maps is extremely flexible. If a property name cannot be found in a data map, connectors assume that it is a repository property name. Hence, you can use a mixture of data map element names and repository property names in a query. Also, if all participating repositories in a MultiQuery object use the same property names, you can execute the MultiQuery without a data map.

Next, you call various setter methods to set search conditions on the query object. You can limit the query by item class, search container, selection criteria, or full text condition. You can also request result set sorting and set the maximum number of results to retrieve. Before calling any of these setter methods, it is a good practice to verify that the repository has the desired capability. The class RepositoryProfile contains a long list of flags to indicate which functions are implemented in a particular connector. Example 8-6 shows how to set the search criteria after verifying that the repository can query properties.

*Example 8-6 Setting selection criteria on a query*

```
// query literals need to be enclosed in single quotes
String criteria = “zipCode IN (‘95123’, ‘95141’, ‘95136’)“;

//make sure the repository supports selection criteria and then set it
if (repo.getRepositoryProfile().getCanQueryProperties()){
   query.setSelectionCriteria(criteria);
}
```

The commandline samples ExecQuery and ExecMultiQuery contain an extensive list of available query options and corresponding RepositoryProfile checks.

Archived

**Tip:** Try to keep your client application code repository-agnostic, so that it does not require much development effort if you need to support an additional repository at a later time:

- You can use MultiQuery even if you currently only need to query a single repository.
- Use the Java API class RepositoryProfile to discover which operations are supported by the current repository instead of hard-coding repository-specific if-else blocks. If a certain capability is not included in the default RepositoryProfile, you can define your own repository profile class that contains custom capability flags.

These guidelines give you the flexibility to add new repositories to the application in the future with minimal code changes.

If you ever need to identify which type of repository you are working with, use the method repository.getRepositoryProfile().getRepositoryType(). Valid repository types are defined as constants in the class com.venetica.vbr.configuration.BridgeConfig.

After setting all search parameters, you can execute the query with the execContentQuery() method, as shown in Example 8-7.

*Example 8-7 Executing a query*

```
IResultSet result = query.execContentQuery();
IResultSet fedResult = fedQuery.execContentQuery();
```

A single-repository query returns an instance of SimpleResultSet, while a federated query returns an instance of ResultSet. Both result set classes implement the IResultSet interface. Note that the class ResultSet offers the method setBucketSize(). Since Version 8.5, this method has no implementation and does not influence the result retrieval any longer.

## The inner workings of MultiQuery

When Access Services receives a MultiQuery request, it determines the repositories that participate in the federated search and dispatches requests to each repository. Each repository query is mapped to a single-repository Query object and runs in a separate thread.

The final result of a federated search is a single aggregated result set that appears as though it came from one system that housed all of the enterprise's content. Each result row contains an item handle with the connector name. Thus, you always know from which repository a particular row originates.

If one of the repository queries results in an exception, for example, due to a time out or an invalid search parameter, the exception is wrapped in a MultiQueryException and stored in the exception array of IResultSet. Other repository queries of the MultiQuery continue uninterrupted.

Federated search can also involve custom pre-processing and post-processing. See 8.3.4, “Federated Query Transformation” on page 429 for more information.

**MultiQuery thread pool**

A thread pooling feature was added to the MultiQuery implementation in Content Integrator 8.5.1. With thread pooling enabled, Access Services saves idle query threads in a pool and reuses them for subsequent queries if they have not timed out, which reduces operating system overhead and improves system stability and performance.

You can see the parameters to configure thread pooling in the Administration Tool in the Access Services Property editor. To enable the feature, set the Enable MultiQuery Thread Pooling flag to **true**. The following parameters have default values. Only enter values to customize the implementation:

- **Maximum thread count** limits the number of active threads in the pool. The default value is -1 (unlimited).
- **Minimum thread count** determines the number of threads to keep in the pool even if they are idle. The default is 0. Enter a value if you have a high performance system and want to reduce the creation of new threads at the cost of never deleting several of them.
- **Inactive thread time out** is the number of seconds after which an inactive thread is terminated and removed from the pool. The default value is 60 seconds. Inactive threads are only terminated if the number of threads in the pool is greater than the minimum thread count parameter.

### 8.2.3 ResultSet operations

A result set consists of an array of ResultRows, an array of ColumnInfos, and an array of Exceptions. The latter contains any exceptions that occurred while executing the query.

If you want a certain order of result rows, you typically set parameters for the result set sorting on the query object before executing the query. Depending on whether the repository supports native sorting, the sorting is performed by the repository or by Content Integrator. If you want a separate row order *after* the query execution, there is an option to perform client-side sorting without issuing a new query to the repository. Client-side sorting is always performed by Content

Integrator in the local JVM. Example 8-8 shows how to perform client-side sorting.

*Example 8-8 Sorting the result set on the client side*

```
// Create a SortSpecification for each column to be sorted by
SortSpecification sortSpec = new SortSpecification(columnName,
SortSpecification.ASCENDING);
// client side sorting
result.sortRows(new SortSpecification[] {sortSpec});

// Display the newly sorted results set
System.out.println(result.toString());
```

You can print the result set through the convenience method ResultSet.toString(). Alternatively, you can loop through the result set yourself and obtain information about each row, as shown in Example 8-9. Note that rows and columns are 0-indexed.

*Example 8-9 Operations on a result set*

```
ResultRow row = null;
ColumnInfo colInfo = null;
RepoItemHandle handle = null;

// loop through each row in the result set
for (int i = 0; i < result.getRowCount(); i++) {
   row = result.getRowAt(i); // 0-indexed

   // obtain general information about the item
   handle = row.getItemHandle();
   String itemName = row.getItemName(); // null if name retrieval off
   String mimeType = row.getMimeType(); // null if mime retrieval off

   // loop through each row's column values
   for (int j=0; j < row.getColumnCount(); j++) {

      // obtain general information about the column
      colInfo = result.getColumnInfoAt(j); // 0-indexed
      String columnName = colInfo.getName();
      int columnDataType = colInfo.getDataType(); // see IDataType

      // obtain column's single or multi-values
      if (row.getContainsMultiValues(j)) {
         String values[] = row.getColumnMultiValues(j);
      } else {
```

```
            String value = row.getColumnValue(j);
        }
    }
}
```

The ColumnInfo class contains the name and data type of each result set column. Data type constants are defined in IDataType.

If a data map property is used as selection property in a federated search and mapped to repository properties of separate data types, the data type of the result set column is IDataType.UNKNOWN. The only exception to this rule is if the repository data types are variants of date (DATE or DATETIME). In that case, the data type of the result set column is DATETIME.

Each ResultRow contains a unique identifier to the corresponding repository item. The identifier is an instance of the class RepoItemHandle. Example 8-10 shows the general format of a RepoItemHandle and two sample handles.

*Example 8-10 Repository item handles*

```
vbr:/Repository ID/Item ID/Version ID/Item Type
vbr:/P8_45/%7BE4C2EDA3-5430-402F-8473-EB48E289BAC0%7D/2.0/CONTENT
vbr:/P8_45/%7B2E28609D-BAF8-4444-8285-E383F7D78D85%7D//FOLDER
```

Both sample handles originate from a connector named P8_45. The first sample handle belongs to a content object. It has the version ID 2.0. If the version number is left blank in a content item handle, the connector interprets that as a request for the latest version. The second handle is a folder handle. Folder handles never have version numbers. The last element of a handle is the item's type (such as CONTENT and FOLDER type). Valid item type values are defined in the interface IItemType. Special characters in RepoItemHandles are URL-encoded.

**Important:** When you finish result set processing, it is important to call the method freeInstance() on the IResultSet object. This action frees resources in Content Integrator and helps prevent a memory leak:

```
// always free-up result sets
fedResult.freeInstance();
result.freeInstance();
```

## 8.2.4 Retrieval of metadata and native content

After you have an item handle, you can use it to retrieve the item's metadata and native content from the repository. To retrieve metadata, call the method getContent() for content items and getFolder() for folder items. The method getRepoItem() is available for items of any type. Example 8-11 shows metadata retrieval.

*Example 8-11 Retrieving content and folder items*

```
// retrieving content metadata (equivalent calls)
RepoItem item1 = repo.getRepoItem(contentHandle);
Content content1 = repo.getContent(contentHandle.getItemID(),
         contentHandle.getVersionID());

// retrieving folder metadata (equivalent calls)
RepoItem item2 = repo.getRepoItem(folderHandle);
Folder folder2 = repo.getFolder(folderHandle.getItemID());
```

Because of the inheritance relationship between Content, Folder, and RepoItem classes, the runtime type of the returned objects is either Content or Folder.

**For more information:** The methods getRepoItem(), getContent(), and getFolder() are comprehensive. They retrieve all of the metadata about the item. See 8.2.9, "Performance tips" on page 409 for how to retrieve selected properties only.

The metadata of an item is represented through the classes Property and PropertyDescription. Example 8-12 shows how to obtain the array of properties of a repo item and the corresponding PropertyDescriptions.

*Example 8-12 Retrieving the properties and property descriptions of a repo item*

```
// loop through each property
Property[] properties = item.getProperties();
for (Property property : properties) {
   String propertyName = property.getName();
   int propertyDataType = property.getType();  // see IDataType
   String[] values = property.getValues();
   if (values.length == 1) {
      // single-value property
   } else if (values.length > 1) {
      // multi-value property
   }
   // obtain the metadata about the property
```

```
    PropertyDescription propDesc =
        item.getPropertyDescriptionByName(property.getName());
    // lots of useful information in the property description, such as
    String propLabel = propDesc.getLabel();
    boolean isSystemProperty = propDesc.getIsSystemProperty();
    boolean canUseAsSelectionCriteria = propDesc.getIsQueryable();
    boolean canUseAsSelectionProperty = propDesc.getIsSelectable();
}
```

The method item.getProperties() returns an array of Property instances. These properties are item class properties. All items of the same item class return the same properties (with separate values, of course). Sometimes, a connector also returns system properties through this method call. System properties are returned for all items, independent of their item class. Examples for system properties are expiration date or document owner. You can use any property that is returned through the getProperties() method as the selection criteria or the selection property in a query or an item finder.

The amount of system properties returned through the getProperties() method varies from connector to connector. Hence, Content Integrator defines a few generic system properties that are always available. These generic properties are available through methods on RepoItem, Content, and Folder objects. Example 8-13 lists a few of them. Generic system properties cannot be used as selection criteria or selection properties in a query or an item finder[2].

*Example 8-13 Generic system properties*

```
String itemName = item.getName();
String itemClassName = item.getItemClassName();
Date creationDate = item.getCreationDate();
Date revisionDate = item.getRevisionDate();
RepoItemHandle handle  = item.getHandle();
// if a content object, cast to Content to get additional fields
Content content = (Content) item;
boolean isCheckedOut = content.getIsCheckedOut();
String fileName = content.getDefaultFileName();
```

To retrieve the native content of an item, use those methods on Repository that start with “getNativeContent”, such as getNativeContentAsFile() or getNativeContentAsStream(). Example 8-14 on page 403 shows how to retrieve the native content of an item as an InputStream.

[2] Item names and Multipurpose Internet Mail Extensions (MIME) types can be retrieved through special flags in queries and item finders.

*Example 8-14 Retrieving the native content of an item*

```
// retrieve the native content as an InputStream
InputStream inStream = repository.getNativeContentAsStream(contentID,
versionID);

// create an output file
File tempFile = new File(fileName);
FileOutputStream outStream = new FileOutputStream(tempFile);

// create a 16K buffer to write data into temporarily
int bufSize = 1024 * 16;
byte[] buffer = new byte[bufSize];
int len = 0;
// read from the input stream and write into the file
// continue reading until number of bytes read is -1
while ((len = inStream.read(buffer, 0, bufSize)) > 0) {
   outStream.write(buffer, 0, len);
   // the number of bytes returned (len) might be smaller than
   // the number of bytes requested (bufSize)
}
// always close the stream, as this frees up connector resources
inStream.close();
outStream.close();

System.out.println("Saved " + tempFile.length() + " bytes to " +
tempFile.getAbsolutePath());
```

**Pages:** Content Integrator supports compound documents. For example, a content item might have a JPG image and a PDF document attached to it. In the Java API, each piece of native content is referred to as a *page*. A page has its own file name and MIME type and can be retrieved separately:

```
int numberOfPages = content.getPageCount();
String pageFileName = content.getDefaultFileNameOfPage(pageNumber);
String pageMimeType = content.getMimeTypeOfPage(pageNumber);
byte[] data = repo.getNativeContentOfPage(..., pageNumber);
```

The term *page* does not represent the number of pages within a text document or operating system pages.

Archived

Content Integrator offers various native content transfer mechanisms between Access Services and the connector. The following list is a mapping of the Java API retrieval method to the transfer mechanism. The same mapping applies to update and checkin methods:

- **Streaming:**
  - `void getNativeContentAsFile(String contentID, String versionID, String fileName)`
  - `void getNativeContentOfPageAsFile(String contentID, String versionID, String fileName, int pageNum)`
  - `InputStream getNativeContentAsStream(String contentID, String versionID)`
  - `InputStream getNativeContentOfPageAsStream(String contentID, String versionID, int pageNum)`
  - `InputStream getNativeContentAsStream(String contentID, String versionID, int chunkSize)`
  - `InputStream getNativeContentOfPageAsStream(String contentID, String versionID, int pageNum, int chunkSize)`
- **Byte Array:**
  - `byte[] getNativeContent(String contentID, String versionID)`
  - `byte[] getNativeContentOfPage(String contentID, String versionID, int pageNum)`
- **HTTP Access:**
  - `String getNativeContentURL(RepoItemHandle itemHandle, String defaultFileName, String mimeType)`
  - `String getNativeContentURLOfPage(RepoItemHandle itemHandle, int pageNum, String defaultFileName, String mimeType)`
  - `HttpAccessInputStream getNativeContentURLAsStream(RepoItemHandle itemHandle, String defaultFileName, String mimeType)`
  - `HttpAccessInputStream getNativeContentURLOfPageAsStream(RepoItemHandle itemHandle, int pageNum, String defaultFileName, String mimeType)`

The information center contains a detailed comparison of the three transport mechanisms. In general, we recommend those methods that use streaming. Streaming returns a standard Java InputStream to client applications. It splits native content into small chunks and sends the chunks separately between the connector and the client, thus, creating a streaming effect. This mechanism allows the transfer of files of any size. For large files, it prevents out of memory errors in the connector and the client JVMs. You configure the chunk size as a

connector parameter in the Administration Tool. The client application can overwrite the chunk size on an individual request basis through a parameter on the getNativeContent..() method.

Native content streaming was introduced in Content Integrator 8.5. In prior versions, the two available mechanisms were Byte Array and HTTP Access.

### 8.2.5 ItemFinder

In addition to the Query and MultiQuery classes, ItemFinder class is a third option to search for and retrieve items and their metadata from a repository. The ItemFinder class is a lightweight retrieval tool that returns all of the items from one or more folders in a repository. The result set is the type IResultSet. Similar to query result sets, the result set of an ItemFinder object contains the RepoItemHandle object for each item in the containers. Optionally, it includes the item names, the MIME types, and the values of selected properties. Example 8-15 shows how to execute an ItemFinder on a folder.

*Example 8-15 Using ItemFinder to retrieve all of the items in a folder*

```
// create ItemFinder
ItemFinder finder = repo.createItemFinder();
finder.addSearchContainer(folderHandle);
// execute ItemFinder
IResultSet resultSet = finder.execute();
```

Use the ItemFinder object to implement a browsable view of a hierarchical repository. Remember that the ItemFinder object does not work recursively; if the contents of a folder and its subfolder need to be retrieved, the ItemFinder object must be called separately on the folder and the subfolder.

In addition to searching folders, ItemFinder allows another use case: A client can specify one or more item handles in the ItemFinder request and leave the search container parameter blank. In this kind of request, the client typically also specifies the selection properties and might also turn on the flags for name and MIME-type retrieval. See Example 8-16 for a code sample.

*Example 8-16 Using ItemFinder to retrieve the metadata of two items*

```
//create ItemFinder
ItemFinder finder = repo.createItemFinder();
finder.addSelectionCriteria(itemhandle1);
finder.addSelectionCriteria(itemhandle2);
finder.addSelectionProperties("firstName,lastName,zipCode");
finder.setIncludeItemNames(true);
```

```
// execute ItemFinder
IResultSet resultSet = finder.execute();
```

In Example 8-16 on page 405, the result set contains two rows: one row for each item handle that was specified as a selection criteria. Also, the result set contains the values of the selected properties firstName, lastName, and zipCode and the item name. With ItemFinder, a client can quickly retrieve selected metadata about items in one request instead of calling the costly getContent() method on each item separately. We describe this performance benefit again in 8.2.9, “Performance tips” on page 409.

It is possible for a client to specify both item handles and search containers. In this case, the result set can contain duplicate items if items are located in a folder and specified again as selection criteria. In contrast to Query, ItemFinder does not allow to filter items in the search containers based on conditions of their property values or full text contents. Currently, you *cannot* specify any result set sorting in an ItemFinder request. However, it is possible to sort the result on the client side after the execution of an ItemFinder through the sort methods on IResultSet.

## 8.2.6 Locale and time zone support

Content Integrator supports deployment in diverse geographies across multiple time zones.

### Client locales

The *client locale* determines the language of the messages that are returned to the client and the language of the labels in the Administration Tool, as well as data formatting and sorting operations. Content Integrator supports 13 languages. To set the client application’s locale, use the Java API method:

```
com.venetica.vbr.client.ClientLocale.setActiveLocale(Locale l)
```

If the client locale is not set, the JVM locale is used, by default.

To print error and warning log messages in a particular language, open the Administration Tool and select a language by clicking **Configuration Data** → **Logging Language**. The language of the log messages is independent of the client locale.

## Handling date and time properties

To simplify the handling of date and time values, Content Integrator represents the values of data types DATE, DATETIME, and TIME in the Java type long:

- For DATE, the long value represents the number of milliseconds since 1 January 1970, 00:00:00 Greenwich mean time (GMT). Only the date portion of the time stamp is relevant. Negative long numbers represent dates before 1970.
- For DATETIME, the long value represents the number of milliseconds since 1 January 1970, 00:00:00 GMT. Both the date and time portions of the time stamp are relevant.
- For TIME, the long value represents the number of milliseconds since midnight local to the time zone in which the property was set.

**Important:** Client applications using the Java API must supply DATE and DATETIME values in GMT. Similarly, all DATE and DATETIME values returned by the Java API are in GMT. For example, if a client wants to update a DATETIME property to `Wed, 10 Sep 2008 14:19:12 PST`, the value must first be converted to GMT (`Wed, 10 Sep 2008 21:19:12 GMT`) and, then, converted into a long (`1221081553000`) value. Then, the long value can be passed to the Java API.

## Time zone support

The individual components of a Content Integrator installation can be distributed across multiple time zones. The client application is responsible for handling the conversion between the user time zone and GMT. The connector takes care of the conversion between GMT and the repository time zone. By default, the connector assumes that it is located in the same time zone as the repository. If that is not the case, you must specify the repository time zone in the connector configuration in the Administration Tool. The parameter is called Time Zone Context.

Example 8-17 shows how to display a DateTime property that is obtained from Content Integrator in a string representation and a target time zone. The example uses the class java.text.SimpleDateFormat to format the long number in GTM as a String in US Eastern Standard Time.

*Example 8-17 Displaying a DateTime property in a target time zone*

```
if (property.getType() == IDataType.DATETIME) {
   // get a DateTime instance of SimpleDateFormatter
   SimpleDateFormat formater = (SimpleDateFormat)
       SimpleDateFormat.getDateTimeInstance(DateFormat.LONG,
           DateFormat.LONG, Locale.US);
```

```
    // convert value from GMT into the time zone expected by user
    formater.setTimeZone(TimeZone.getTimeZone("America/New_York"));
    value = formater.format(Long.valueOf(propValue));
}
```

You can use similar code to display the properties of type date or time. Use the methods getDateInstance() or getTimeInstance() on SimpleDateFormat instead.

We recommend that you always use a property's long value and convert it to a String representation, as shown in Example 8-17 on page 407.

### 8.2.7 API versus SPI

You might notice that certain Java classes contain both API and SPI operations. For example, the class RepoItem contains the API method getHandle() and the SPI method setHandle(). The method getHandle() is used by clients to obtain the unique identifier of an item. In contrast, the method setHandle() is used by the connector to set the unique identifier on the item before returning the item to the client.

**Never use SPI methods in client code:** When writing a client application using Eclipse along with its Content Assist feature (which lists all methods on an object), be aware that this utility shows both API and SPI methods. *Never use SPI methods in client code.* Always hover with the mouse over the method to verify the JavaDoc group tag. Avoid methods that are marked with the JavaDoc tags `@group spi` or `@group internal`. SPI and internal methods are not intended for client use and can lead to unexpected behavior. Methods with the JavaDoc tag `@group beaninfo` are acceptable to use as long as they are not marked `spi` or `internal`, as well. Another way to discover whether a method belongs to the API is by looking at the HTML Javadoc documentation in the *`ICI_HOME`*`/docs/integrate/api` directory.

It might be obvious that the RepoItem.setHandle(handle) method is an SPI method, but there are other SPI methods that can be easily mistaken for API methods, such as Content.setMimeType() or RepoItem.addProperty(). It is not the client's responsibility to set the MIME type on an item. The MIME type is determined by the repository and set by the connector. Similarly, a client does not *add* properties to an item. A client only *sets values* on item properties. The properties of an item are determined by its item class and other factors. When in doubt about the correct usage of methods from a client perspective, check the Java sample code.

## 8.2.8 Non-supported operations

This section contains a list of operations that you cannot perform through any of Content Integrator's APIs. Although the product supports a wide variety of operations to access and modify content, the following administrative operations are *intentionally* not supported:

- You cannot create, modify, or delete item classes. Content Integrator only exposes item classes as they are defined in the repository. A repository administrator with native repository tools defines the item classes.
- You cannot create, modify, or delete property definitions. Again, Content Integrator only exposes property definitions as they are configured in the repository. A repository administrator with native repository tools defines the properties and assigns them to item classes.
- You cannot create, update, or delete users and groups. Content Integrator only exposes users and groups as they are defined in the repository. A repository administrator with native repository tools defines the users and groups.

You can discover more Content Integrator design principles in 6.6, "General design principles" on page 314.

## 8.2.9 Performance tips

Content Integrator is a thin layer between the client application and the repository. Connectors translate a client's request into one or more repository API calls and then send the repository response back to the client. It is a design principle of Content Integrator to leave request validation and processing to each repository. Therefore, the performance overhead of Content Integrator is small. The major factors that determine the performance of a content integration solution are the available system resources, the performance of the back-end repositories, and the geographical location of deployed components (network latencies).

A client application can impact the system's performance through the number and type of API methods that it calls. The following list contains general performance tips and describes tasks that can be accomplished through multiple alternative Java API methods:

- Metadata retrieval: The methods getRepoItem(), getContent(), and getFolder() on the class Repository return *all* metadata about an item, including its item class properties, system properties, and security information. If you are only interested in the item class properties, name, or MIME type of an item or a group of items, we recommend that you specify the requested properties during the Query or ItemFinder execution, or use

ItemFinder with item handles as the selection criteria (see Example 8-16 on page 405).

- Native content retrieval: If you want to retrieve only the native content of an item but not its metadata, use one of the getNativeContent...() methods on Repository instead of those methods on Content. Both sets of methods return the same data. However, with the methods on Repository, you avoid the overhead of populating the Content object. Also, for Content Integrator 8.5 and later, we recommend that you use the native content streaming methods instead of the deprecated Byte Array or HTTP Access methods. Native content streaming supports files of any size and keeps the memory footprint of the client and connector low.
- Folder browsing: The Folder class uses a special caching mechanism that has performance implications. The class contains two internal caches: one cache with Content objects and one cache with Folder objects. If populated, the items in the two caches represent all of the items that are located in this folder. The following methods on Folder make use of the internal caches: getContentCount(), getContent(index), getSubFolderCount(), and getSubFolder(index). Before Folder determines the content count or returns an item of a particular index, it first retrieves all Content objects (or all Folder objects) from the repository into its local cache and then executes the request from the cache. This approach is advantageous if a client frequently retrieves items from the same folder and wants to minimize the calls between the connector and the repository. However, populating items is an expensive operation. Folder might be slow if there are many items in the folder and the client only wants to determine the item count or retrieve a few of the items. In those cases, ItemFinder and Repository are faster alternatives. You can use ItemFinder and Repository in the following ways:
  - To determine the number of items in a folder, use ItemFinder with a search container instead of Folder.getContentCount().
  - To determine the number of subfolders in a folder, use ItemFinder instead of Folder.getSubFolderCount(). Because you cannot instruct an ItemFinder to only return folders, the client needs to loop through the result set and filter out folders by comparing the item type of each row handle against the constant IItemType.FOLDER.
  - To retrieve a particular item from a folder, use Repository.getContent(id) or Repository.getFolder(id) instead of Folder.getContent(index) or Folder.getSubFolder(index). This approach assumes that you know the item ID already.
- Name and MIME type retrieval: When running an ItemFinder or a Query, the flags includeNames and includeMimeTypes are false, by default. On certain repositories, it is an expensive operation to retrieve this information. Only set these flags to true if you need to retrieve item names or MIME types.

- Queries: When performing a query, limit the result set by specifying search conditions, an item class, a search container, or a max result. This limit reduces the number of items to search in the repository and improves the query execution time. Also, make sure that the repository is set up properly and tuned. Content Integrator queries can only be as fast as the repository search itself. Contact IBM support or Lab Services for assistance with tuning IBM repositories.
- Logins: To reduce the time that it takes to establish a repository connection, consider the usage of session pools. See 8.3.1, “Session pools” on page 412 for more details.
- Retrieval of large result sets: Query and ItemFinder result sets do not offer a cursoring mechanism. All of the rows of the result set are transferred in one batch from the repository through the connector to the client application. If you need to retrieve a large number of items, you can implement a batching algorithm in the client application. This batching algorithm limits memory consumption and increases the number of users that the system can support at the same time. For a client batching algorithm, you need a property whose values are unique in order to split the result set into smaller sets based on the values of the unique property. Depending on the scope of the items to search, “unique” can apply to an item class, folder, or the repository. Content Integrator offers the data type IExtendedDataType.UNIQUE_ID to mark repository properties that are known to be unique. Use the following query approach:
  - Issue the first query using these search options:
    selection property: `unique property`
    selection criteria: `unique property LIKE *`
    order by: `unique property ascending`
    max result: batch size
  - Issue all subsequent queries using these search options:
    selection property: `unique property`
    selection criteria: `unique property >= last value from previous result set`
    order by: `unique property ascending`
    max result: batch size
  - Keep using this query until there are no more results left.

For performance tips that are related to the deployment of connectors and other Content Integrator components, see 6.6, “General design principles” on page 314.

# 8.3 Advanced Java API topics

In this section, we take a closer look at six advanced Java API topics:

- Session pools
- Single sign-on
- Configuration API
- Federated query transformation
- Security models
- Complex properties

## 8.3.1 Session pools

Establishing a repository connection and authenticating can be expensive operations in certain content management systems. Content Integrator session pooling helps to reduce that logon time by reusing existing idle repository connections instead of creating new repository connections.

The Content Integrator session pool component provides these abilities:

- Reuse repository connections within a client application or among several client applications.
- Limit the repository connections that are consumed by a given client application during peak activity.
- Prevent the leaking of repository login connections by client applications that terminate unexpectedly.

The major component of the session pool implementation is the session pool server. The session pool server manages User objects in a pool. A User object contains all of the repository connections that are opened by a particular client. The session pool server creates User objects on demand, logs the objects into the requested repositories, and returns the objects to client applications. The server tracks the connections and can limit the number of concurrent connections to a particular repository. It reuses or times out released User objects and determines which User to return for a given client request. Multiple client applications can reuse the repository connections in a User object if the applications use the same repository credentials. The session pool server insures that the User object is checked out by only one client at a time.

### A sample session pool scenario

We look at a sample session pool scenario and walk through the process of obtaining a User object with a repository connection. Table 8-2 on page 413 shows the session pool cost parameters. The cost parameters are relative.

*Table 8-2 Sample session pool cost parameters*

| Cost name | Cost relative value |
|---|---|
| Relative cost to create users | 1 |
| Relative login cost, IBM FileNet P8 | 20 |
| Relative logout cost, IBM FileNet P8 | 10 |
| Relative login cost, Documentum | 15 |
| Relative logout cost, Documentum | 10 |

Figure 8-2 shows a session pool server with two client applications and a pool with three User objects.

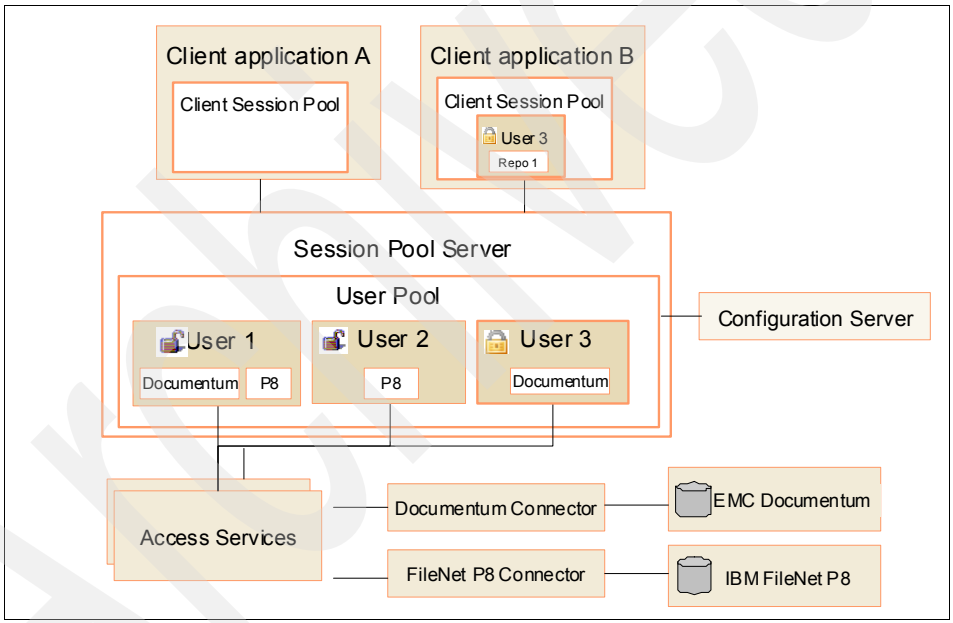



*Figure 8-2 Sample session pool scenario*

User 1 in the pool has open connections to Documentum and IBM FileNet P8. This user is currently idle and can be checked out by a client application. User 2 has an open connection to FileNet P8 and is idle, as well. User 3 has an open connection to Documentum and is in use by client application B. This user is locked and cannot currently be checked out by another client.

Assume that client application A requests a User object for Documentum. The session pool server calculates the reuse cost for all idle users in its pool, compares it against the cost to create a new user, and logs it into Documentum. Table 8-3 shows the costs for each option.

*Table 8-3 Sample session pool cost calculation*

| Option | Cost components | Total cost of option |
|---|---|---|
| Create a new User | Create new user cost: 1<br>Login cost Documentum: 15 | 16 |
| Reuse User 1 | Logout cost FileNet P8: 10 | 10 |
| Reuse User 2 | Logout cost FileNet P8: 10<br>Login cost Documentum: 15 | 25 |
| Reuse User 3 | Not possible, because the User is checked out | N/A |

User 1's connection to Documentum can only be reused if the credentials that are provided by client application A are the same as the credentials of User 1. In this example, we assume that the credentials match, that is, that the AuthBundle field's user name, password, and optional are the same. If the credentials do not match, the cost to log out User 1 from Documentum and the cost to log in that user to Documentum (with separate credentials) need to be added to the overall cost of reusing User 1. User 1 also has an open connection to FileNet P8. This connection must be closed, because client application A did not provide credentials for FileNet P8. Therefore, the cost to reuse User 1 is 10, which is the logout cost for FileNet P8.

User 2 needs to be logged into Documentum and logged out of FileNet P8. The total cost for reusing User 2 is the sum of these two costs (25).

As we can see in the Total cost of option column in Table 8-3, User 1 has the lowest total costs (10). The session pool server checks out User 1 from the pool, closes its connection to FileNet P8, and returns it to client application A.

You can set session pool limitations with the parameters *Maximum total logins*, *Maximum logins per user id*, *Maximum maintained logins*, and *Minimum maintained logins* per repository. For a detailed explanation of all session pool parameters, consult the information center.

An update to the parameter values in the Administration Tool is automatically picked up when the session pool server "polls" the Configuration Server for updates. No session pool server restart is required. You can configure the polling interval by using a JVM property or set it in the session pool constructor. See "Local and remote session pool servers" on page 415 for details. A client

application can force a non-scheduled update by invoking the updateConfig() method on the class SessionPool.

**Relative costs:** When you create a new session pool configuration in the Administration Tool, the cost parameters and the min/max restrictions have default values. We strongly recommend that you customize these values according to your requirements and the actual costs in your environment. The three cost parameters are relative. They do not have a unit of measure. For example, if the relative login cost is set to 10 and the relative logout cost is set to 1, it means that logging into the repository takes 10 times as long as logging out of it. The same rule applies to cost comparisons across repositories in one session pool configuration.

Most often, the cost factors correspond to the time that is required to log in and log out of a particular repository, but the user is free to modify the factors based on other considerations. For example, if a particular repository is known to “leak” connections, the user can adjust the cost factors higher to encourage the reuse of those connections in order to slow the leak.

The cost to create a new User object is set to 1 by default. The default value is extremely low due to the fact that the initialize() operation on a User object executes quickly. This operation used to be more expensive in Content Integrator 8.4 and prior versions when Access Services was located on an application server and had to be initialized with a Java Naming and Directory Interface (JNDI) call.

To follow the cost calculation of the session pool server, enable debug logging in the session package. Example 8-18 sets the logging level to DEBUG and directs output to the appender A1.

*Example 8-18 Enable DEBUG logging in the session pool server*

```
log4j.category.com.venetica.vbr.client.session=DEBUG,A1
```

## Local and remote session pool servers

In a distributed environment, the session pool server can be launched and maintained in a separate JVM. This type of deployment is called a *remote* or *distributed session pool server*. Alternatively, the session pool server can run in the same JVM as the client application, which is referred to as a *local session pool server*. Figure 8-3 on page 416 displays the two setups and lists the session pool constructors that correspond to each setup.

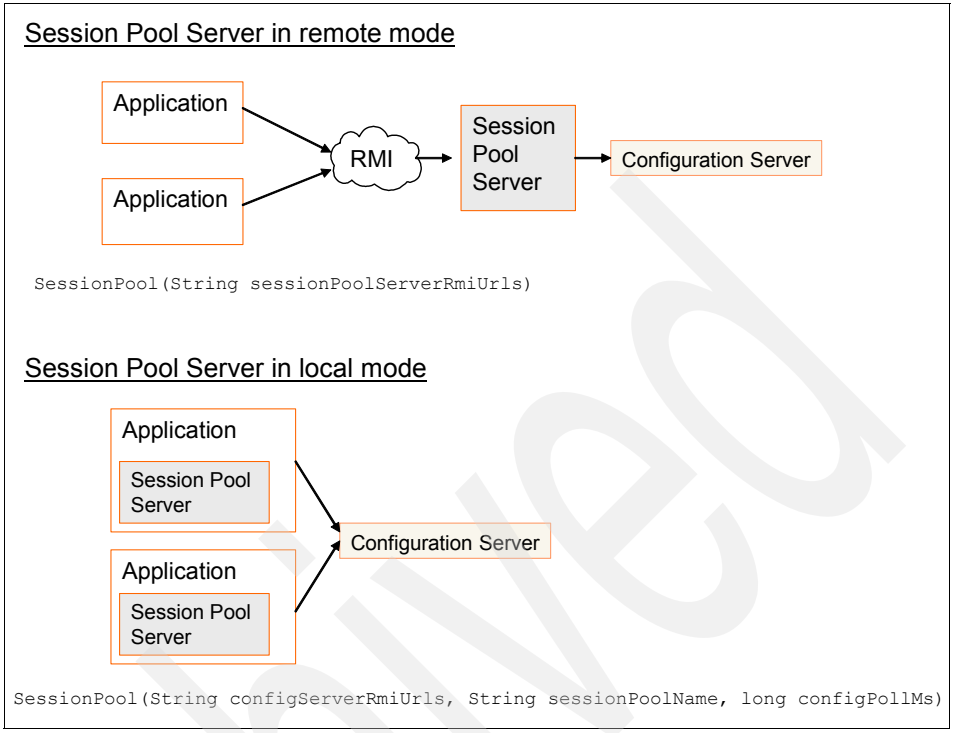



*Figure 8-3 The session pool server in local and remote modes*

The advantage of a remote session pool server is that multiple client applications can access the same server and share repository connections among each other. The drawback of a remote session pool server is that User objects must be serialized and sent over RMI between the client application and the session pool server. Credential encryption is supported. We show an example next.

In contrast, a session pool server in local mode is an application-only server that eliminates the remote call to obtain User objects. Each application controls the life cycle of its session pool server independently from other applications. Consequently, sharing repository connections across client applications is not supported in local mode. You can set up sharing repository connections across client applications with custom client code.

Example 8-19 on page 417 shows the client code for using a session pool server in remote mode. The session pool object created in line one represents the client view of a given session pool. This client-side session pool object does not contain the actual pool with User objects. It only stores a list of User references that are checked out by this client application.

*Example 8-19 Client code for obtaining a User from a remote session pool server*

```
// instantiate a Session Pool given a remote Session Pool Server
SessionPool pool = new SessionPool("rmi://spServer:1250/MyPool");

// create credentials for two repositories
RepositoryLoginCredential[] userCreds = new
   RepositoryLoginCredential[2];
userCreds[0] = new RepositoryLoginCredential("P8_45", "usr", "myPass");
userCreds[1] = new RepositoryLoginCredential("CM8", "admin", "pwd1");

// obtain a User object for these two repositories
User repoUser = pool.obtainUser(userCreds);

   //do stuff

// release the User object back to the pool
pool.releaseUser(repoUser, false);

// When done with the pool, close it.
pool.close();
```

**Important:** Never call freeInstance() on a User object obtained from a session pool. Only *release* the User back to the session pool. The session pool server might reuse the User session, and the session pool server handles releasing resources when appropriate.

Alternatively, you can obtain a User from the session pool server using encrypted AuthBundle. Example 8-20 shows the client code for this scenario. The session pool server compares the client's AuthBundle with the AuthBundle of idle User objects. The Session pool Server can determine a match even if the bundles are encrypted. The session pool server does not need the Blowfish encryption key. Only the connector unseals the AuthBundle and, hence, needs the same encryption key as the client application.

*Example 8-20 Obtaining a session pool user with a sealed AuthBundle*

```
// create an AuthBundles
AuthBundle authBundleP8 = new AuthBundle("usr", "myPass");

// Instantiate a BlowfishSealer
BlowfishSealer bS = new BlowfishSealer();

try {
   // Encrypt the auth bundle
```

```
    bS.seal(authBundleP8);
} catch (EncryptionException e) {
    e.printStackTrace();
    return;
}

// create a credentials object for the repository
RepositoryLoginCredential[] userCreds = new
   RepositoryLoginCredential[1];
userCreds[0] = new RepositoryLoginCredential("P8_45", authBundleP8);

// obtain a User object for the repository
User repoUser = pool.obtainUser(userCreds);
```

You can start a session pool server in one of two ways:

1. Start the SessionPoolServerLauncher with the command that is shown in Example 8-21.

   *Example 8-21 Starting a SessionPoolServer through its launcher program*

   ```
   java -Dvbr.home=<path to Content Integrator home> -classpath=vbr.jar
   com.venetica.vbr.client.session.SessionPoolServerLauncher
   ```

2. Execute the script VeniceBridgeServices.bat (Windows) or VeniceBridgeServices.sh (UNIX/Linux) in the *ICI_HOME*/bin directory. This script calls the SessionPoolServerLauncher in addition to the launchers of the other services.

   **Note:** In Content Integrator 8.5 or later, the file VeniceBridgeServices.sh or VeniceBridgeServices.bat prints messages that might need explanation:

   - "WARN [VBr] (ITLMUtil) COE00524W: The license request was denied because the Tivoli License Manager agent is not installed or other internal issues". You can safely ignore this warning message if you do not use Tivoli License Manager.
   - "Required property vbr.services.registryport is not defined." You can disregard this warning message. It prints even if the property is defined correctly.
   - "Session pool server Session Pool 1 listening on port 0." Port number 0 prints incorrectly. The actual port number is either the value of vbr.services.registryport or the port number that is defined in vbr.session.pool.urls.

Archived

The SessionPoolServerLauncher is a stand-alone program that binds one or more session pool server instances and makes them available through RMI. The launcher program requires a Configuration Server URL from which to obtain the pool configuration. The location of the Configuration Server must be specified with a JVM property. There are other optional JVM properties. Specify the command line option -help to display all of the JVM properties of the launcher program. You can set these JVM properties as command line options or in the *ICI_HOME*/vbr_services.properties file. Table 8-4 lists all of the JVM properties.

*Table 8-4 JVM properties of the SessionPoolServer Launcher program*

| JVM property | Description |
|---|---|
| vbr.configuration.server .lookup.urls | This JVM property is a required property that defines a comma-delimited list of RMI URLs for the Configuration Server. For example: rmi://fortress:1099/ConfigSvr, rmi://backup:1099/ConfigSvr. You do not need to have Configuration Server running. You can also specify the location of the configuration file with the file://... URI format. |
| vbr.session.pool.names | This JVM property is an optional property that defines a comma-delimited list of session pools to start. If the property is omitted, all configured session pools are started. |
| vbr.session.pool.urls | This JVM property is an optional property that defines a comma-delimited list of session pool URLs. The number of URLs in the list must be the same as the number of pools being started. If the property is omitted, session pool URLs are determined by the name of the session pool. |
| vbr.configuration.server .polling.period.ms | This JVM property is an optional property that specifies how often the Configuration Server (or configuration file) must be polled for an updated session pool configuration. The value is represented in milliseconds. The default is one minute. |

In local mode, a separate session pool server is not required. Example 8-22 on page 420 shows the client code for using a session pool server in local mode. This time, the Java constructor expects three parameters:

- The location of the Configuration Server. The location can be specified with an RMI URL (pool1) or by pointing to the location of the configuration file (pool2).
- The name of the Session Pool, as defined in the Administration Tool.

Archived

- The polling interval determines the frequency with which the local session pool server checks with the configuration server or the configuration file for updates to the session pool configuration.

As in the remote scenario, the session pool objects pool1 and pool2 represent the client view of a given session pool. They only contain those User objects that are checked out by the client application. The remaining code of how to obtain a User from a session pool and how to release the User is identical to the code that is shown for the remote scenario.

*Example 8-22 Client code for obtaining a User from a local session pool server*

```
// instantiate a local Session Pool Server using a Configuration Server
SessionPool pool1 = new
SessionPool("rmi://host:1250/ConfigurationServer", “Session Pool 1”,
60000);

// alternative way using config.xml
String configLocation = "file:///" + System.getProperty("vbr.home") +
"/config.xml";
SessionPool pool2 = new SessionPool(configLocation, “Session Pool 2”,
60000);

RepositoryLoginCredential creds = new RepositoryLoginCredential(“CM8”,
“icmadmin”, “pwd”);

// obtain a User object
User repoUser = pool1.obtainUser(new RepositoryLoginCredential[]
{creds});

   //do stuff

// release the User object back to the pool
pool1.releaseUser(repoUser, false);
// When done with the pool, close it.
pool1.close();
```

**JVM1.5:** When using IBM JVM 1.5, client applications that are connected to a session pool server in local mode do not finish completely after closing the pool and exiting. This defect in the IBM JVM is known. A system thread hangs in suspended mode indefinitely. The defect does not have any effect on the session pool functionality. All Content Integrator resources are cleaned up correctly.

Archived

## 8.3.2 Single sign-on

Authentication in a federated system is more complex than in a single repository system due to the system's inherent heterogeneity. Each participating repository can have its own authentication mechanism and user accounts. A client application, which provides a single user interface to separate back-end repositories, must determine how to handle federated authentication. For example, a federated search against three repositories requires three pairs of credentials, one pair for each repository. Keeping track and entering credentials to many systems can be cumbersome for the user.

With Content Integrator Single Sign-On (SSO), a client application can give users a seamless experience and hide the federated nature of the system. This transparency is achieved through a secure credential vault, which contains the repository login information for each registered user. A user only needs to log in one time to the SSO system. If authenticated, the client application retrieves repository credentials from the credential store and handles authentication to the back-end repositories transparently for the user. Content Integrator refers to the users of an SSO system as *subjects*.

Figure 8-4 displays the basic flow in an SSO authentication. First, user Alice logs in by providing her subject user name and password. Content Integrator authenticates the subject by comparing the credentials with those credentials that are stored in the SSO credential store. If they match, the client application can look up repository credentials stored for subject alice in the credential store and use them to log into back-end repositories on behalf of Alice.

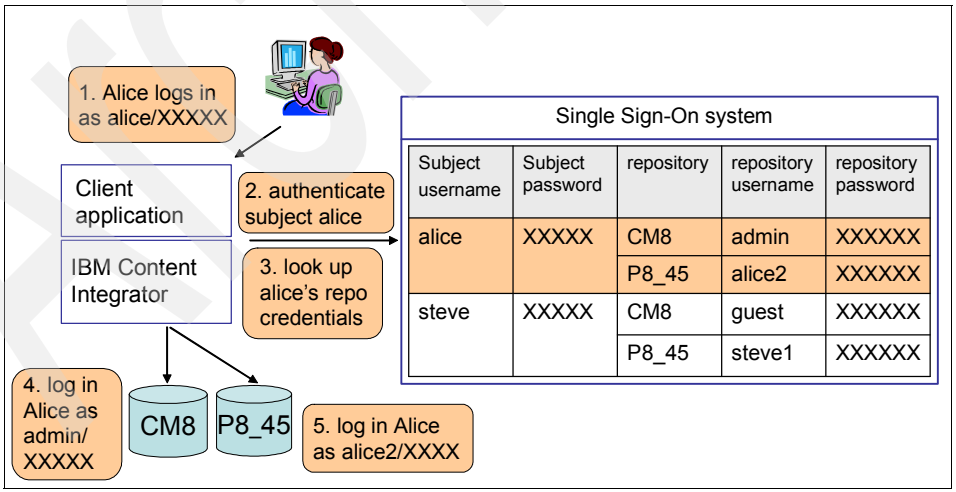



*Figure 8-4 Basic Single Sign-On flow*

Archived

Content Integrator SSO offers a Service Provider Interface (SPI) that allows you to plug in a credential store of your choice, such as an LDAP directory or a database. There are two data store implementations included with Content Integrator: an XML file store and a plug-in to a Lightweight Directory Access Protocol (LDAP) directory (deprecated). The XML file store is the reference implementation and enabled by default.

## Single Sign-On Server

SSO implementations can have complex configuration requirements that might require cryptography libraries and keys to access the authentication system. To allow client applications easy access to those authentication systems, Content Integrator provides a *Single Sign-On Server*. A Single Sign-On Server functions as a proxy between the client application and the SSO implementation. The client application only needs to access the Single Sign-On Server instead of the authentication system. The Single Sign-On Server must be configured to access the authentication system. This way, client development is simplified and independent of authentication system details. See Figure 8-5.

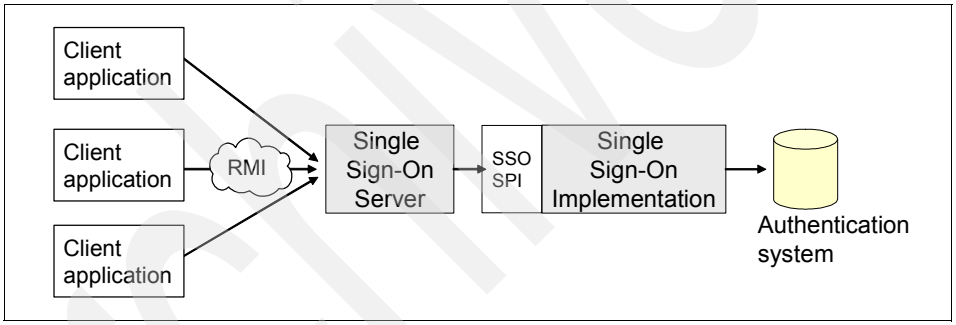



*Figure 8-5 Single Sign-On Server*

You can start the Single Sign-On Server or SSO server in one of two ways:

- Execute the script VeniceBridgeServices.bat (Windows) or VeniceBridgeServices.sh (UNIX or Linux) in the *ICI_HOME*/bin directory. This script calls the SSOServerLauncher in addition to the launchers of the other services. See SessionPoolServerLauncher in “Local and remote session pool servers” on page 415 for details about the launcher startup script.
- Call the SSOServerLauncher program directly (Example 8-23).

*Example 8-23 Starting a Single Sign-On Server through its launcher program*

```
java -Dvbr.home=<path to Content Integrator home> -classpath=vbr.jar
com.venetica.vbr.sso.server.SSOServerLauncher
```

The Single Sign-On Server is configured through various JVM properties. All properties have default values that are defined in the *ICI_HOME*/vbr_services.properties configuration file. Table 8-5 lists the JVM properties that are used by the Single Sign-On Server.

*Table 8-5 JVM properties of the Single Sign-On Server*

| JVM property | Description |
|---|---|
| vbr.sso.server.rmiurl | This JVM property is a single RMI URL to be associated with the SSO server. The default value is rmi://localhost:1250/SSOServer. |
| vbr.sso.authenticator | This JVM property is the fully qualified class name of the SSO SPI implementation. The default value is com.venetica.vbr.sso.refsso.ReferenceAuthenticator. You can change that value to com.venetica.vbr.sso.ldap, LdapAuthenticator, or to your own custom SSO authentication implementation. The named class must implement the interface IAuthenticator. |
| vbr.configuration.server.lookup.urls | This JVM property is a list of comma-separated URLs of Configuration Servers where connector information can be retrieved, for example: rmi://myserver1:1250/ConfigurationServer,rmi://myserver2:1250/ConfigurationServer. Alternatively, you can specify the location of the configuration file with the file://.. URI format. The SSO server needs to look up the mapping of the connector names to persistent connector names, because it allows the usage of both types of connector identifiers. |
| vbr.sso.server.checkForUpdatesInterval | This JVM property is the frequency in milliseconds with which the SSO Server checks the configuration for updates. The default is 1 minute. Use 0 for always, a negative value for never. |
| vbr.sso.server.encrypt | This JVM property is the no-op property; it is not used in the code. |

Example 8-24 on page 424 shows a code sample of how to connect to the Single Sign-On server and to retrieve repository credentials.

Archived

*Example 8-24 Single Sign-On authentication client code*

```
//Create the subject's AuthBundle to authenticate to the SSO Service
AuthBundle ab = new AuthBundle(subjectName, password);

// bundle can be optionally be encrypted. See Example 8-2 on page 392

// Connect to the SSO Server through the SSOService class
SSOService sso = new SSOService("rmi://localhost:1250/SSOServer");

//Retrieve the SSO Subject
ISSOSubject subject = sso.retrieveCredentials(ab);

//Retrieve the subject's credentials for a particular repository
RepositoryLoginCredential cred =
   subject.getRepositoryLoginCredential(connectorName);

//Retrieve the repository AuthBundle
AuthBundle repoBundle = cred.getAuthBundle();

// Create and initialize the user.
User user = new User();
user.initialize();
// Retrieve the specified repository
Repository repo = user.getRepositoryByID(connectorName);

// Log into the repository.
repo.encryptedLogon(repoBundle);
System.out.println("Logged in! Logon ID is " + repo.getLogonID());

// Logoff.
repo.logoff();
// Free User
user.freeInstance();
```

You can find the complete example in the Java commandline sample `SSOLogon.java` file in the *`ICI_HOME`*`/docs/examples/java/commandline` directory.

## The SSO reference implementation

The reference SSO system is provided as a default implementation of the SSO SPI and is enabled by default. It uses an XML-based data store, but it allows other persistence mechanisms to be plugged in through the SSO Persistence SPI. For more information about the SSO Persistence SPI, consult the SPI Javadocs and the Information Center.

The reference SSO implementation is configured through various JVM properties. All of the properties have default values that are defined in the *ICI_HOME*/vbr_services.properties configuration file. Table 8-6 lists these JVM properties.

*Table 8-6 JVM properties for the reference SSO implementation*

| JVM property | Description |
|---|---|
| vbr.sso.server.vbr.crypto.digest .class | This property is the fully qualified digest class to be used by the encryption algorithm. The default value is com.venetica.vbr.crypto.MD5Digest. |
| vbr.sso.server.vbr.crypto.cipher .class | This property is the fully qualified cipher class to be used by the encryption algorithm. The default value is com.venetica.vbr.crypto.BlowfishCipher. |
| vbr.sso.server.vbr.datastore .class | This property is an optional property that is used by the SSO persistence manager. Default value: com.venetica.vbr.sso.refsso.XMLDataStore. Only set this value if you are plugging in a separate credential store implementation. The named class must implement the ISSODataStore interface. |
| vbr.sso.server.vbr.datastore .xml.file | This property is an optional property that is used by the XMLDataStore class to find the XML file. If specified, it must contain the full path and file name of the XML file to use. The default file is called sso.xml in the directory specified by vbr.home. |
| vbr.sso.authenticator .listeners | This property is an optional property that contains a comma-separated list of listeners to the SSO system. Each class must implement the IAuthenticatorListener interface. The SSO system informs listeners about add, remove, and update events and gives listeners the right to veto. |

You can manage subjects and their repository credentials with the interactive Java sample commandline.SSOAdminTool. The SSO administrator tool (SSOAdminTool) needs write access to the authentication system; therefore, it does not interact with the Single Sign-On Server. It connects directly to the reference implementation.

**Single Sign-On migration tool**
If you have subjects and repository credentials that are stored in the SSO reference implementation in Content Integrator versions 8.4 or 8.3 and want to upgrade to 8.5, you need to migrate this data. Content Integrator 8.5 uses an XML-based file store instead of the former FastObjects database. There is a Single Sign-On migration tool that converts your data into the correct XML format. For details about how to run the tool, see the Information Center.

### 8.3.3 Configuration API

The Configuration API provides client applications the ability to programmatically access and modify the configuration of an Content Integrator installation. The API supports commonly used operations that are offered by the Administration Tool, such as creating, importing, and exporting connectors, data maps, and session pools. The Configuration API makes integrating and embedding Content Integrator in other products more seamless, because configuration can be performed programmatically and in the background.

A Content Integrator configuration can be described as a tree of configuration data. The tree or individual tree components can be exported and stored persistently as an XML file. These XML files can later be imported. The XML export and import feature is useful for test automation or during a version upgrade of Content Integrator. It lets you recreate an entire set of configurations in one simple operation.

The Configuration API supports the following operations:

- Connectors:
  - Import and export a connector.
  - Create a connector from scratch and modify connector properties.
  - Add and remove a connector from an Content Integrator configuration.
  - Get a connector from an Content Integrator configuration.
  - List available connector types to add to an Content Integrator configuration.
- Data maps:
  - Import and export a data map.
  - Add and remove a data map from an Content Integrator configuration.
  - Get a data map from an Content Integrator configuration.
- Session pools:

  - Import and export a session pool.
  - Add and remove a session pool from an Content Integrator configuration.
  - Get a session pool from an Content Integrator configuration.
- Configuration tree:
  - Import and export an Content Integrator configuration.
  - Create a new configuration tree.

The classes of the Configuration API reside in the com.venetica.vbr.configuration package. The major classes in that package are ConfigurationTree, BridgeConfig, DataMapConfig, and SessionPoolConfig. Example 8-25 shows how to import and export a Content Integrator configuration.

*Example 8-25 Importing and exporting an Content Integrator configuration*

```
// import configuration tree from the config server
String url = "rmi://host:1250/ConfigurationServer";
ConfigurationTree configTree = ConfigurationTree.loadFromServer(url);

// import configuration tree from the config file
File configFile = new File(System.getProperty("vbr.home") +
   File.separator + "config.xml");
ConfigurationTree config = ConfigurationTree.loadFromFile(configFile);

// perform modifications

// export configuration tree to the config server
configTree.saveToServer(url);
// export configuration tree to a file
configTree.saveToFile(configFile);
```

Example 8-26 shows how to create a new connector configuration for the IBM Content Manager OnDemand repository and to test the connection. All of the parameters of the connector configuration are available through the Configuration API. The connector-specific parameters, such as ODServer, are provided in a Java properties container. These parameters have the same default values as a new Content Manager OnDemand connector configuration in the Administration Tool.

*Example 8-26 Creating a new connector configuration programmatically*

```
// create a new configuration (alternatively import one)
ConfigurationTree configTree =
   ConfigurationTree.createConfigurationTree();
```

```
// create a new connector configuration
BridgeConfig connectorConfig =
   BridgeConfig.createConnector(BridgeConfig.IBM_ODWEK_CONNECTOR);

// set general connector parameters
connectorConfig.setDescription("CMOD Test Repo");
connectorConfig.setName("CMOD");
connectorConfig.setPersistentID(BridgeConfig.createPersistentID());

// set connector specific parameters
// the property names are provided with default values, now customize
Properties properties = connectorConfig.getProperties();
properties.setProperty("ODServer", "9.30.216.103");
properties.setProperty("ConfigDir", "C:\\Program Files\\IBM\\ODWEK");
properties.setProperty("RMIPort", "1445");

// set RMI Proxy Connector Server parameters
connectorConfig.setUseRmiProxyConnector(true);
connectorConfig.setExternalURL("rmi://9.30.216.122:1271/RMIBridgeServer
");

// set logging parameters
LoggingInfo loggingInfo = new LoggingInfo();
loggingInfo.setIsErrorLoggingEnabled(true);
loggingInfo.setIsWarningLoggingEnabled(true);
loggingInfo.setIsTraceLoggingEnabled(false);
loggingInfo.setStdoutLogging(true);
connectorConfig.setLoggingInfo(loggingInfo);

// add new connector config to the ConfigurationTree
configTree.addConnector(connectorConfig);

// test connection
User user = new User();
user.initializeFromConfig(configTree);
Repository repo = user.getRepositoryByID("CMOD");
repo.logon(username, password, optional);
System.out.println("Logged on with logon ID "+ repo.getLogonID());
repo.logoff();
user.freeInstance();
```

Additional sample code is available in the Java commandline sample files DumpConfigurationTree.java, ImportConnectorConfiguration.java, and RepoTestVirtualConfig.java.

### 8.3.4 Federated Query Transformation

In an environment where multiple repositories store related data, logically similar item properties can have diverse formatting and, thus, a federated query can return unexpected results. For example, repositories might store the postal code property in various ways, as shown in Example 8-27.

*Example 8-27 Format of the property postal-code in three repositories*

```
repository1: postal_code = 28208
repository2: postal_code = 282085210
repository3: postal_code = 28208-5210
```

Assume that a user issues a federated query where `postal_code = "28208-5210"`. The user might expect content from all three repositories to be included in the result set. However, only results for repository3 are returned.

With Content Integrator's *Federated Query Transformation* feature, you can tweak attribute values and other aspects of the query prior to query execution. You can also perform a post-transformation on the result set.

Continuing with the previous example, the transformation code can perform the property value adjustments that are described in Example 8-28 prior to query execution.

*Example 8-28 Sample transformation on postal_code property values*

- For repository1: update value to use first 5 chars only: use 28208
- For repository2: update value to remove the dash: use 282085210
- For repository3: make no transformations to value: use 28208-5210

In this case, the result set provides the desired outcome for the user.

You implement federated Query Transformation through an SPI that provides a extension point into the general flow of multiquery processing. The transformation code must implement the IFederatedQueryTransformer interface in the com.venetica.vbr.ejb.spi package. This interface contains the five methods preProcessMultiQuery(), preProcessQuery(), postProcessResults(), postProcessResultSet(), and destroy(). Example 8-29 shows the sequence of these method calls in pseudo code.

*Example 8-29 Federated Query Transformation pseudo code*

```
MultiQuery processing is initiated
     user's SPI implementation instantiated using no-arg constructor
```

```
     preProcessMultiQuery(MultiQuery) is called
         for each repository in MultiQuery
              preProcessQuery(Query) is called
              Query processing occurs…
              postProcessResults(Query, QueryResults) is called
         end for loop
      postProcessResultSet(ServerResultSet) is called
      destroy() is called
MultiQuery processing is complete
```

Your custom transformer must implement all five methods that are defined in IFederatedQueryTransformer. However, not every method must necessarily perform an action. It is valid to print a log statement or return the input parameter object without any modification. You can use the destroy() method to perform cleanup operations. Due to the thread-based execution of federated queries, the order in which preProcessQuery() and postProcessQuery() are called for individual repositories is non-deterministic.

To enable custom transformation code, place the executable into the *ICI_HOME*\spi directory and enter the fully qualified class name in the Administration Tool under the Access Services section in the field titled Federated Query Transformer Class. For more details, see the Information Center. An example implementation exists in the following directory:

*ICI_HOME*\docs\examples\java\spi\federatedQueryTransformation

### 8.3.5 Security models

The Java API lets you retrieve and modify permissions on individual repository items. To account for differences in repository security models and client requirements, there are four available security models:

- Simple security model
- Advanced security model
- Enhanced security model
- Read access token security model

Table 8-7 on page 431 lists which connector implements which security model. Note that specific connectors implement multiple security models and certain connectors are restricted to read-only operations within a security model.

Archived

*Table 8-7 Connectors and their security models*

| Security model | Connectors implementing the model |
|---|---|
| Simple security | IBM FileNet Image Services |
| Advanced security | IBM FileNet P8<br>IBM FileNet Content Services<br>IBM WebSphere Portal Document Manager<br>IBM Content Manager 8 (read-only)<br>IBM Lotus Notes (read-only)<br>IBM Domino Doc (read-only)<br>Microsoft NT File system<br>Documentum<br>Hummingbird Document Management<br>Open Text Livelink<br>Teamsite (read-only) |
| Enhanced security | IBM Content Manager<br>IBM FileNet Content Services<br>IBM FileNet P8<br>Documentum<br>Open Text Livelink |
| Read access token security | IBM FileNet P8<br>IBM FileNet Content Services<br>IBM WebSphere Portal Document Manager<br>Microsoft Windows SharePoint Services<br>Documentum<br>Hummingbird Document Management<br>Open Text Livelink |

You can use the following methods on the RepositoryProfile class to determine programmatically which connector implements which security models:

```
getUsesSimpleSecurityForContent()
getUsesSimpleSecurityForFolders()
getUsesAdvancedSecurityForContent()
getUsesAdvancedSecurityForFolders()
getUsesEnhancedSecurity()
getUsesSecurityTokens()
```

Each security model is associated with specific classes and methods in the Java API. The following section defines the capabilities of each model and lists the associated API classes and methods.

Archived

## Simple security model

The simple security model supports assigning one principal to each of the three predefined privileges: read, write, and delete. The principal can be a user or a group. Table 8-8 lists all of the Java API methods of the simple security model. The methods on ContentBase apply to both content and folder items, because ContentBase is the super class of Content and Folder. Example 8-30 shows a code sample.

*Table 8-8 Java API classes and methods of the Simple security model*

| Security model | Java API classes and methods |
|---|---|
| Simple security | **Methods on ContentBase**<br>void setSimpleCanReadPermission(String userName)<br>void setSimpleCanUpdatePermission(String userName)<br>void setSimpleCanDeletePermission(String userName)<br>int getPermissionsCount()<br>Permissions getPermissions(int index)<br>**Methods on Permissions**<br>boolean getCanRead()<br>boolean getCanUpdate()<br>boolean getCanDelete()<br>String getName()<br>int getType() |

*Example 8-30 Retrieving and updating permissions with the simple security model*

```
// assign each permission to one principal
content.setSimpleCanReadPermission(“MyUser”);
content.setSimpleCanWritePermission(“MyOtherUser”);
content.setSimpleCanDeletePermission(“AdminGroup”);

// save changes in the repository
content.updateSecurity();

// read content again to re-populate fields
content = repository.getContent(content.getContentID(),
Content.LATEST_VERSION);

// print out permissions
for (int i = 0; i < content.getPermissionsCount(); i ++) {
   Permissions permissions = content.getPermissions(i);
   System.out.println(“Principal: “ + permissions.getName());
   System.out.println("User or group?: " + permissions.getType());
   System.out.println(“Can Read?: “ + permissions.getCanRead());
   System.out.println(“Can Update?: “ + permissions.getCanUpdate());
```

```
   System.out.println("Can Delete?: " + permissions.getCanDelete());
}
```

You can obtain a list of repository user and group names through methods on Repository: getUserCount(), getUser(index), getGroupCount(), and getGroup(index).

## Advanced security model

The advanced security model supports the concept of an access control list (ACL) with the four predefined privileges: read, update, delete, and change security. Each of these privileges can be assigned to one or more users or groups per item. Table 8-9 lists the Java API methods that belong to the advanced security model. The methods on ContentBase apply to both content and folder items, because ContentBase is the super class of Content and Folder.

*Table 8-9 Java API classes and methods of the Advanced security model*

| Security model | Java API classes and methods |
|---|---|
| Advanced security | **Methods on ContentBase**<br>void addCanReadPermission(String userName,boolean canRead)<br>void addCanUpdatePermission(String userName,boolean canUpdate)<br>void addCanDeletePermission(String userName,boolean canDelete)<br>void addCanChangeSecurityPermission(String userName,boolean canChangeSecurity)<br>int getPermissionsCount()<br>Permissions getPermissions(int index)<br>**Methods on Permissions**<br>boolean getCanRead()<br>boolean getCanUpdate()<br>boolean getCanDelete()<br>boolean getCanChangeSecurity()<br>String getName()<br>int getType() |

Example 8-31 shows sample code of how to use the advanced security model to retrieve and update permissions on an item. Note that the methods to retrieve permissions are identical between the simple and advanced security model: both security models use the Permissions class.

*Example 8-31 Retrieving and updating permissions with the advanced security model*

```
// assign each permission to one or more principals
content.addCanReadPermission("MyUser", true);
content.addCanReadPermission("MyOtherUser", true);
content.addCanReadPermission("MyThirdUser", true);
```

```
content.addCanUpdatePermission(“MyUser”, false); // denies permission
content.addCanDeletePermission(“MyOtherUser”, true);
content.addCanChangeSecurityPermission(“MyThirdUser”, true);

// save changes in the repository
content.updateSecurity();

// read content again to re-populate fields
content = repository.getContent(content.getContentID(),
Content.LATEST_VERSION);

// print out permissions
for (int i = 0; i < content.getPermissionsCount(); i ++) {
   Permissions permissions = content.getPermissions(i);
   System.out.println(“Principal: “ + permissions.getName());
   System.out.println("User or group?: " + permissions.getType());
   System.out.println(“Can Read?: “ + permissions.getCanRead());
   System.out.println(“Can Update?: “ + permissions.getCanUpdate());
   System.out.println(“Can Delete?: “ + permissions.getCanDelete());
   System.out.println(“Can Change Security?: “ +
permissions.getCanChangeSecurity());
```

You can obtain a list of repository user and group names through methods on Repository: getUserCount(), getUser(index), getGroupCount(), and getGroup(index).

## Enhanced security model

The enhanced security model is the most elaborate security model of the Java API (see Table 8-10 on page 435). It exposes repository security features, such as ACLs, users, groups, group memberships, and document owners. It offers a choice between four predefined privileges (federated privileges) and native repository privileges. To learn more about the enhanced security model, see the information center.

Archived

*Table 8-10 Java API classes and methods of the enhanced security model*

| Security model | Java API classes and methods |
|---|---|
| Enhanced security | Methods on ContentBase:<br>AccessControlList getAccessControlList()<br>void setAccessControlList(AccessControlList acl)<br>void replaceAccessControlList(AccessControlList acl)<br>void changeOwner(RepoPrincipal newOwner)<br><br>All classes in package `com.venetica.vbr.client.security`:<br>AccessControlList<br>AccessControlEntry<br>RepoSecuritySchema<br>RepoPrivilege<br>FederatedPrivilege<br>PrivilegeSet<br>RepoUser<br>RepoGroup and others |

Example 8-32 shows how to grant document read permission to a user through the methods of the enhanced security model.

*Example 8-32 Retrieving and updating permissions with the enhanced Security model*

```
// retrieve security schema, which contains users, groups, and more
RepoSecuritySchema schema = repository.getSecuritySchema();
IRepoPrincipal myUser = schema.getRepoUser("MyUser");

// create a "federated" read privilege
Privilege federatedReadPriv =
schema.getFederatedPrivilege(FederatedPrivilege.FEDERATED_READ);
// out of curiosity, check which repo privileges it maps to
Privilege[] repoReadPrivs= federatedReadPriv.getRepoPrivileges();

PrivilegeSet privSet = new PrivilegeSet(new Privilege[]
{federatedReadPriv});

// retrieve the Access Control List for this item
AccessControlList acl = content.getAccessControlList();
// create a new entry that allows myUser read operations
AccessControlEntry entry = acl.createAccessControlEntry(myUser,
   privSet, Disposition.ALLOW);
acl.addEntry(entry);

// persist security changes to the repository
```

```
content.updateSecurity();

// retrieve content again before proceeding
content = repository.getContent(content.getContentID(),
Content.LATEST_VERSION);

// retrieve the updated ACL
acl = content.getAccessControlList();

// get all entries for the user and its groups
Iterator iterator = acl.getEntries(myUser, true);

// retrieve details about each ACL entry
while (iterator.hasNext()) {
   AccessControlEntry entry = (AccessControlEntry) iterator.next();
   String principalName = entry.getPrincipal().getName();
   FederatedPrivilege[] fedPrivs =
       entry.getPrivilegeSet().getFederatedPrivileges();
}
```

The Java commandline samples InteractiveAcl and ExploreUsersAndGroups contain additional sample code for the enhanced security model.

**To persist permission changes to the repository:** Call the method updateSecurity() on ContentBase. This method covers the simple, advanced, and enhanced security models. Do not use the methods ContentBase.update() or Content.updateWithNewNativeContent*(), because those methods only update metadata and native content, but not security information. After the updateSecurity() call, retrieve the item again so that fields are populated according to the new permissions.

## Read access token security model

The read access token security model exposes access tokens representing entities that have native content access for a given document. An entity can be a user, a group, or a role. As shown in Example 8-33 on page 437, access is granted if there is an intersection between the tokens of a user and the tokens of the requested document.

*Example 8-33 Retrieving permissions with the read access token security model*

```
String[] contentTokens = content.getReadAccessTokens();
// for example: ["g:research", "g:engineering", "u:mblack"]

String[] userTokens = repository.getReadAccessTokensForUser("MyUser");
// for example ["g:public", "u:jsmith", "g:engineering"]

if (intersection (contentTokens, userTokens).size() > 0 ) {
   // access granted
} else {
   // access denied
}
```

The token-based security model was developed to support search engines that crawl content repositories, in particular IBM OmniFind® Enterprise Edition. Table 8-11 lists the Java API methods of the read access token security model.

*Table 8-11 Classes and methods of the read access token security model*

| Security model | Java API classes and methods |
|---|---|
| Read access token security | ContentBase.getReadAccessTokens()<br>Repository.getReadAccessTokensForUser(String userName)<br>ItemFinder.getCheckNativeContentAccess()<br>ItemFinder.setCheckNativeContentAccess(boolean flag) |

## 8.3.6 Complex properties

Content Integrator exposes the metadata of a repository item as a set of properties. The corresponding classes in the Java API are Property and PropertyDescription. A property has a data type and a single or multiple values. Example 8-34 shows a multivalue property.

*Example 8-34 A multivalue property of data type String*

```
Authors=Sunil, Vikram, Anita (type=STRING)
```

Repositories can have more complex property structures that cannot be adequately expressed with the Property class. The Java API offers the classes ComplexProperty and ComplexPropertyDescription to support nested properties of diverse data types.

A complex property has the data type IDataType.COMPLEX. Its value is an array of complex property instances. An instance contains one or more properties,

each of which is either simple or complex. There is no limit to the depth of nesting or to the number of properties per nesting level.

Example 8-35 shows a sample complex property called Insured. The complex property has two instances, each of which contains a set of three simple properties.

*Example 8-35 A complex property with two instances*

```
Insured (type=COMPLEX)
   Insured[0]
      Insured/InsrdFName=Alexander (type=STRING)
      Insured/InsrdLName=Churchill (type=STRING)
      Insured/YearOfBirth=1947 (type=SHORT)
   Insured[1]
      Insured/InsrdFName=Joan (type=STRING)
      Insured/InsrdLName=Churchill (type=STRING)
      Insured/YearOfBirth=1984 (type=SHORT)
```

The name of a simple property within a complex property contains all names of its parent properties. It follows the pattern:

*<nameOfComplexProperty>/...<nameOfNestedComplexProperties>...*
*/<nameOfSimpleProperty>*

One example is Insured/InsrdFName.

## Connector support

Complex properties were introduced to support child properties in IBM Content Manager 8. The IBM DB2 Content Manager 8 connector is currently the only connector that supports complex properties.

## Retrieving complex properties

Example 8-36 on page 439 shows how to extract values from a complex property. Use the method getComplexValues() to obtain the instance array from a complex property. Do not use the methods getValue() or getValues(). These methods are inherited from the super class Property and return `null` if they are executed on a complex property.

*Example 8-36 Traversing a ComplexProperty*

```
private void printComplexProperty(Property prop) {

      if (prop.getType() == IDataType.COMPLEX) {

         // cast property into a complex property
         ComplexProperty complexProp = (ComplexProperty) prop;

         // extract the instance values into an array
         ComplexProperty[] valueArray = complexProp.getComplexValues();

         // loop through the instance value array
         for (int i = 0; i < valueArray.length; i++) {

            // each instance value itself is of type ComplexProperty
            ComplexProperty instance = valueArray[i];
            int propCount = instance.getPropertyCount();

            System.out.println(complexProp.getName() + "["+i+"]:");

            // loop through the properties of a value instance
            for (int j=0; j < propCount; j++) {
               Property childProp = instance.getProperty(j);
               if (childProp.getType() == IDataType.COMPLEX) {
                  printComplexProperty(childProp);
               } else {
                  System.out.print(childProp.getName() + "=");
                  String[] values = childProp.getValues();
                  if (values.length == 1) {
                     // single value property
                     System.out.print(values[0]);
                  } else {
                     // multi value property
                     for (int k=0; k < values.length; k ++) {
                        System.out.print(values[k] + " ");
                     }
                  }
                  int dataType = childProp.getType();
                  // use helper class to display the data type name
                  System.out.println(" (" +
                      IDataType.Util.toString(dataType) + ")");
               }
            }
            System.out.println("");
```

```
        }
      }
    }
```

Please note that both the complex property and each instance in its instance array are of the ComplexProperty class and have the IDataType.COMPLEX data type.

The traversal of complex property descriptions is simpler, because a complex property description does not have instances. It only describes the metadata that applies to all of the instances of a corresponding complex property.

*Example 8-37 Traversing a ComplexPropertyDescription*

```
private void printComplexPropertyDescription(PropertyDescription
                                             propDesc) {

  if (propDesc.getDataType() == IDataType.COMPLEX) {

    // cast property description into a complex property description
    ComplexPropertyDescription complexPropDesc =
      (ComplexPropertyDescription) propDesc;

   System.out.println(complexPropDesc.getName() + ":");
   int childCount = complexPropDesc.getPropertyDescriptionCount();

    // loop through the child property descriptions
    for (int i = 0; i < childCount; i++) {

      PropertyDescription childPropDesc =
        complexPropDesc.getPropertyDescription(i);

      if (childPropDesc.getDataType() == IDataType.COMPLEX) {
        printComplexPropertyDescription(childPropDesc);
      } else {
        System.out.println(" " + childPropDesc);
      }
    }
    System.out.println("");
  }
}
```

## Creating complex properties

Next, we look at the creation of complex properties. To create a complex property instance, follow this process:

1. Obtain a complex property from the repo item using the method repoItem.getPropertyByName(name) or repoItem.getProperty(index).
2. Obtain the corresponding complex property description from the repo item using the method repoItem.getPropertyDescriptionByName(name) or repoItem.getPropertyDescription(index).
3. Create a complex property value instance by calling the method createComplexPropertyValue(complexPropDesc) on the complex property that was obtained in step 1. Pass the complex property description as the method parameter.
4. Loop through the properties of the complex property that was obtained in step 3 and set values on them.
5. Repeat steps 3 and 4 for each complex property instance to be added. Store each instance in an array. When you have finished, set the array on the complex property using the method setComplexValues().

This process is the only way to create complex property instances. Do not be tempted to use seemingly similar methods, such as the constructor of ComplexProperty, addProperty(property), or setValues(String[]). These methods are SPI methods, and they do not produce a new complex property.

*Example 8-38 Creating a ComplexProperty*

```
public void setValuesOnComplexProperty(ComplexProperty complexProp,
                        ComplexPropertyDescription complexPropDesc,
                        int numberOfInstances)
                        throws VeniceBridgeException, RemoteException{

  ComplexProperty[] complexPropertyValueArray = new
     ComplexProperty[numberOfInstances];

  // one loop for each complex property instance
  for (int j=0; j < numberOfInstances; j++) {

    ComplexProperty complexPropValue = (new
      ComplexProperty()).createComplexPropertyValue(complexPropDesc);
    int propCount = complexPropValue.getPropertyCount();

    for(int i=0; i<propCount; i++)  {

      PropertyDescription childPD =
```

Archived

```
        complexPropDesc.getPropertyDescription(i);
      Property childProp =
        complexPropValue.getPropertyByName(childPD.getName());

      // complex property? recursive method call
      if (childPD.getDataType() == IDataType.COMPLEX) {
        ComplexPropertyDescription complexPD =
          (ComplexPropertyDescription)childPD;
        ComplexProperty complexProperty = (ComplexProperty)childProp;
        setValuesOnComplexProperty(complexProperty, complexPD,
          numberOfInstances);
      } else {
        // simple property: set single or multi-value
        String simpleValue = ""; // your code here to create a value
        childProp.setValue(simpleValue);
      }
    }
    // store each new instance in an array
    complexPropertyValueArray[j] = complexPropValue;
  }
  // when all instances are created, set them on the complex property
  complexProp.setComplexValues(complexPropertyValueArray);
}
```

## Updating complex properties

Finally, we look at the update scenario. If you want to update any aspect of a complex property, use the following approach:

1. Obtain the complex property through the method getPropertyByName(name) or getProperty(index) on RepoItem.
2. Extract the array of complex property instances and save the array in a variable, as shown in Example 8-36 on page 439.
3. Update the instance array as necessary. You can add, remove, or modify complex property instances:
   a. To add an instance, use the createComplexPropertyValue()method, as shown in Example 8-38 on page 441. Create a new array that is one element larger, and copy the existing instances, as well as the new instance, into the new array.
   b. To delete an instance, create a new smaller array and copy all instances into the new array, except for the instance that you want to delete. Do not set an instance to null if you want to delete it.

Archived

c. To modify the value of a simple property within a complex property, navigate to the simple property as shown in Example 8-36 on page 439. Modify the value using the Property.setValue(newValue)method.

4. Set the updated instance array on the complex property from step 1 using the complexProp.setComplexValues(complexPropertyValueArray) method. It is not possible to update only a particular complex property instance. You must always set the whole instance array.
5. Call update() on the content or folder item. To look at the updated version of the item in the repository, you must reload the item with Repository.getRepoItem().

## 8.4 Web Services

The Java API of Content Integrator is an extremely comprehensive programming interface and the default recommendation for new client applications. However, it is not suitable for every type of client application. For example, non-Java clients cannot directly call a Java interface. They need to translate calls from their language to Java, which is a considerable effort. For another example, client applications that are located outside of a company firewall might be precluded from accessing a connector or repository that is located inside the intranet because of firewall security restrictions.

For these kinds of clients, Content Integrator offers a Web Services interface. Web Services are based on open XML standards and are not bound to a particular programming language. Hence, client applications of any language can issue a Web Services request. The transport mechanism for Web Service request and response messages is HTTP. These messages can cross a firewall without problems, because HTTP ports are usually open.

Content Integrator offers two sets of Web Services:

- Web Services API
- Service-oriented architecture (SOA) Web Services

These two Web Services interfaces differ in the scope and granularity of services that they offer. The *Web Services-API* offers 68 services, many of which map one-to-one with methods in the Content Integrator Java API. It follows the remote procedure call paradigm. The services are stateful: a client needs to establish a repository connection through the login Web Service before it can call any of the other services. Each subsequent request must contain a connection ID to identify the client. The Web Services API is based on Apache Axis 2. It uses SOAP over HTTP as the communication protocol between the Web service client and server.

Archived

*SOA Web Services* follow a message-oriented paradigm. There are only 15 services, each of which is broader in scope and maps to multiple operations in the Java API. There is no concept of a session between two Web service calls. Each request is stateless and must include user credentials. SOA Web Services are based on Apache Axis2 and also use SOAP as the communication protocol. SOA Web Services were introduced in Version 8.5. They are easier to use than the Web Services API and are the recommended product from now on for any new implementation.

Figure 8-6 displays the Web Services communication flow. This flow applies to both SOA Web Services and Web Services API.

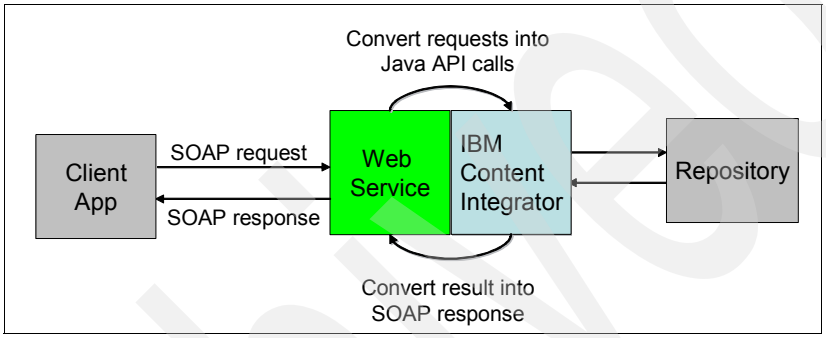



*Figure 8-6 Web Services communication flow*

## 8.4.1 Web Services Description Language

Web Services specify details about request and response messages in the Web Services Description Language (WSDL), which is a standard XML format. A service definition in WSDL represents a contract between the Web service client and server. It defines service names, input and output parameters, and the data types of the parameters. Table 8-12 lists the locations of the WSDL files of the two Content Integrator Web Service interfaces.

*Table 8-12 Location of WSDL files*

| Web Services interface | Location of WSDL file |
|---|---|
| SOA Web Services | `ICI_HOME/soa/SOAService.wsdl` |
| Web Services API | `ICI_HOME/WebServices/WebServicesAPI.wsdl` |

In addition, the WSDL of the SOA Web Services is available in browsable HTML format in the directory `ICI_HOME/docs/wsdl-docs`.

**Using the WSDL file:** The WSDL file is a key element for developing a Web Service client. Given a WSDL file, you can use a Web Service framework, such as Apache Axis2 or Microsoft .Net, to generate corresponding stub classes. Stub classes perform low-level tasks, such as sending and receiving SOAP messages and converting between Java and XML.

Because of the support of Web service frameworks and the simplicity of using stub classes, Web service client development is considered easy and offers a short time to value.

### 8.4.2 Services

SOA Web Services offer the following 15 services:

- ExecuteRepositorySearch
- ExecuteFederatedSearch
- GetFolderItems
- RetrieveDocument
- RetrieveFolder
- CreateDocument
- CreateFolder
- UpdateItem
- DeleteItem
- GetRepositoryList
- GetAccessibleRepositoryList
- GetDataMapList
- GetDataMapDefinition
- GetItemClassList
- GetItemClassDefinition

The `SOAService.wsdl` file and its HTML representation both contain an extremely detailed description of each of these services. You can find a short description of each service in the information center at:

http://publib.boulder.ibm.com/infocenter/ce/v8r5/topic/com.ibm.discovery.ci.appdev.doc/cdnap024.htm

The Web Services API offers 68 Web Services. You can find the complete list of services in the `WebServicesAPI.wsdl` file and the parameter types in the `WebServicesAPI_schema1.xsd` file.

Currently, neither of the two Web services interfaces covers all of the Java API's capability. However, the SOA Web Services are continually being enhanced with more functionality. To learn whether a particular feature is supported, check the WSDL file, ask in the Content Integrator online forum, or contact IBM support.

Archived

## 8.4.3 Setting up the environment

In this section, we explain how to set up your environment.

### Web Service client

For both sets of Web Services, Content Integrator provides sample Java client stubs. The stubs were generated with the Apache Axis2 tool wsdl2java. Table 8-13 lists the location of the client stubs for the two interfaces.

*Table 8-13 Location of the sample Java client stubs*

| Web Services interface | Location of sample Java client stubs |
|---|---|
| SOA Web Services | *ICI_HOME*/soa/SOAService-test-client.jar |
| Web Services API | *ICI_HOME*/docs/examples/wsapi/java/com/ibm/eci/wsapi |

In this section, we assume that the Web Service client application uses these stub classes. The setup for clients in other languages is similar to the steps that we describe.

You need to add the following classes to the client application's classpath:

- Add the stub classes that are mentioned in Table 8-13. For the SOA Web Services, simply add the jar file to the classpath. For the Web Services API, add the *ICI_HOME*/docs/examples/wsapi/java directory to the classpath.
- Add all of the Apache Axis2 1.4.1 jars.

Optionally, if you want to use credential encryption between the Web Service client and server, you need to have the same BlowfishKey.ser file on both machines. The JVM property vbr.home must be set on both client and server JVMs and point to the directory that contains the encryption key file.

A Web Service client application requires the vbr.jar JVM property to use the default Blowfish encryption implementation. For example, the SOAService-test-client.jar file contains the helper class EncryptionUtil.java, which can create a SOAP encryption header and perform credential encryption. This helper class requires the vbr.jar JVM property in the client classpath. A Web Service client application does not need to have the config.xml file.

Archived

### Web Service server

The server components of the Content Integrator Web Services are packaged as Web Application Archives (WARs). Table 8-14 lists the location of the WAR files for each interface.

*Table 8-14 Location of Web Service WAR files*

| Web Service interface | Location of Web service WAR files |
|---|---|
| SOA Web Services | *ICI_HOME*/war/vbr_soa.war |
| Web Services API | *ICI_HOME*/war/vbr_wsapi.war |

Each WAR file is a self-sufficient archive that does not require any additional libraries. It contains a Web application version of Apache Axis2 1.4.1, the Content Integrator Web services, server side stubs, and the vbr.jar JVM property. Deploy this WAR file to an application server, such as IBM WebSphere Application Server or Apache Tomcat.

After deployment, add the vbr.home JVM property to the Web application server. The property must point to the directory where the `config.xml` configuration file is located. If credential encryption is used, the `BlowfishKey.ser` file must be located in that same directory, as well. A full installation of Content Integrator is not required on the Web server machine, as long as these two files are available.

The information center contains detailed instructions for deploying on IBM WebSphere Application Server. Follow these instructions, because they contain important steps about how to modify the class loader sequence:

http://publib.boulder.ibm.com/infocenter/ce/v8r5/topic/com.ibm.discovery.ci.appdev.doc/cdnap019.htm

You can verify a successful deployment by opening the Axis2 welcome page at `http://IP_ADDRESS:PORT_NUMBER/CONTEXT_ROOT`. The context root is set during the deployment phase. On the welcome page, there is a link titled Services. Click this link, and verify that all services are running. The Service ERP parameter contains the URL that client applications need to provide to locate the Web service.

## 8.4.4 Samples

Table 8-15 on page 448 lists the location of the sample programs for each interface.

*Table 8-15 Location of Web Services samples*

| Web Service interface | Location of sample files |
|---|---|
| SOA Web Services | *ICI_HOME*/docs/examples/soa |
| Web Services API | *ICI_HOME*/docs/examples/wsapi/java/wsclient<br>*ICI_HOME*/docs/examples/wsapi/Microsoft.net |

To run the SOA Web Services samples, use the `run_soa_sample.bat` batch file (Windows) or the `run_soa_sample.sh` script file (UNIX or Linux). Before running the script for the first time, open it and enter a value for the variable *AXIS2_HOME*. The script needs the location of the folder that contains all Axis2 jars, because these jars have to be added to the sample's classpath.

If *AXIS2_HOME* contains a space in its path, enclose the variable *lcp* in double quotation marks:

```
SOA_CLASSPATH=%SOA_CLIENT%;"%lcp%"
```

To run the SOA SearchRetrieveSample, call the commands that are shown in Example 8-39.

*Example 8-39 Running an SOA Web Services sample*

```
cd C:\Program Files\IBM\ContentIntegrator\docs\examples\soa
run_soa_sample sample.SearchRetrieveSample
```

The Web Services API samples do not have a script to run them. Consequently, to run them, you need to prepare the classpath yourself and add all of the Axis2 jars and the client stub folder. For example, to run the Web Services API Java sample RepoTest, use the commands that are shown in Example 8-40.

*Example 8-40 Running a Web Services API sample*

```
cd C:\Program Files\IBM\ContentIntegrator\docs\examples\wsapi\java
java -classpath %AXIS2_JARS%;. wsclient.RepoTest
```

In Example 8-40, the Axis2 jars are included with the help of an environment variable. The client stub folder is included through the dot. If you find adding the Axis2 jars too tedious, you can use the run_soa_sample and copy and modify it.

Like the Java API samples, all Web Services samples require input parameters, such as the Web Service URL, user name, password, and so on. To discover which parameters apply to a particular sample, run the sample without any parameters.

9

# Developing and extending connectors

IBM Content Integrator provides a single, uniform, Java-based, bidirectional interface to any number of disparate content repositories. This single interface makes it easy for application developers to integrate all of their information sources into new or existing enterprise applications. Currently, Content Integrator supports many common content repositories. If you want to integrate repositories that do not have supported Content Integrator connectors, you can build a custom Content Integrator connector. In this chapter, we describe how you can implement a new connector or extend an existing connector to add or restrict connector functions.

We cover the following topics in this chapter:

- Implementing a Content Integrator connector
- Extending the existing Content Integrator connector
- Connector plug-ins
- Summary

This chapter serves as a starting point for connector development. If you require more assistance, contact IBM Lab Services or your IBM marketing representative.

© Copyright IBM Corp. 2010. All rights reserved.

# 9.1 Implementing a Content Integrator connector

Before proceeding further with this chapter, review the list of supported connectors to see if your repository is already supported and if an extended connector is necessary:

http://www.ibm.com/support/docview.wss?rs=86&uid=swg27015930

In this section, we cover the following steps to create a brand new Content Integrator connector:

1. Planning and designing your new connector
2. Implementing your new connector
3. Building, deploying, and configuring your new connector

## 9.1.1 Planning and designing your new connector

Perhaps the most important piece of connector development is the planning and design phase. As a connector developer, if you do not have an understanding of how your repository matches up with Content Integrator, the development phase can take longer than normal.

### Understanding your repository

Content Integrator provides a Connector Feasibility Study document to further assist a connector author in the development of a connector. You can download this document at the following location:

ftp://ftp.software.ibm.com/software/dw/dm/db2/dm-0512graciano/connector_feasibility_study.doc

There are several concepts about your repository that are important to understand and that are discussed in that document. The connector developer must analyze the repository to determine what it is storing, as well as how it is organized. In this section, we discuss the important concepts that are necessary so that you will understand the terminology: content, folder, metadata, and unique identifier. We also discuss the considerations to which you need to pay close attention.

#### *Content*

Content Integrator can handle native content, metadata, or both. It is important to understand what your repository contains prior to creating the connector. When developing the connector, it is necessary to distinguish between content and metadata and to allow the consumer of the connector to discover which types of data are available.

Many repositories have unique ways of storing data, making it difficult to distinguish between native content and metadata. A general rule is that if the data is unstructured (such as files or images), it is considered *native content*, as opposed to structured data (that might describe the content), which is considered *metadata*.

Native content is not always associated in a one-to-one relationship with a single item in a repository. In certain repositories, a single content item can have multiple pieces of native content (called *pages* in Content Integrator). Items can also have links to other items. Link relationships can follow a parent-child pattern (called *multipart content* in Content Integrator), or they can simply be named (represented by properties with an extended datatype of INTERNAL_ATTACHMENT or EXTERNAL_ATTACHMENT in Content Integrator). Items that are linked to can also have their own native content, metadata, and links to other items. If your repository uses multipart content, your connector implements the getMultiPartNativeContent() method. Figure 9-1 shows the example scenarios.

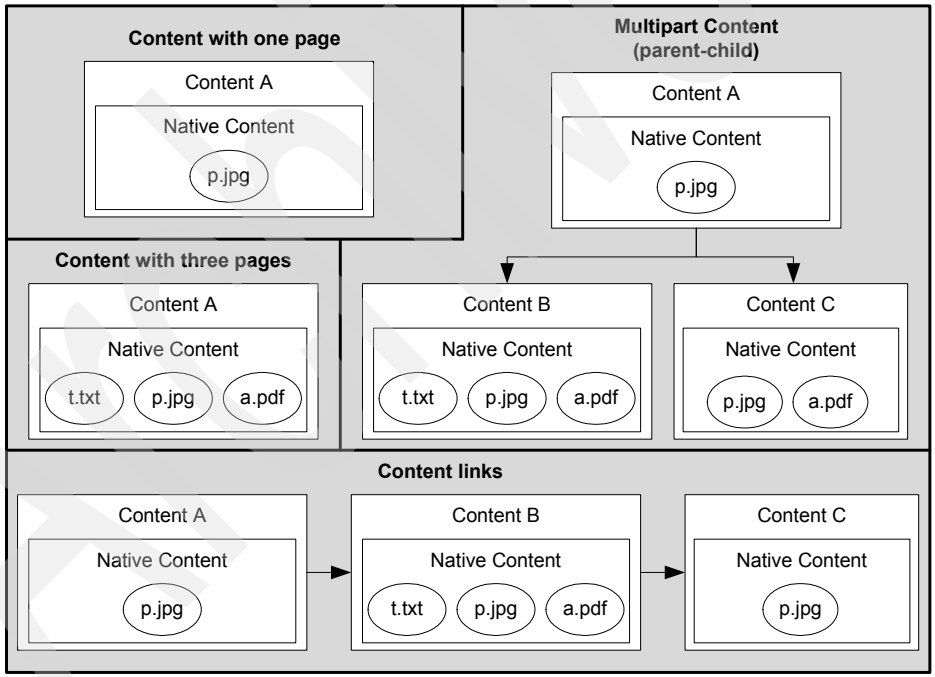



*Figure 9-1 Content scenarios*

The last point to consider regarding your repository's content is whether it specifies the Multipurpose Internet Mail Extensions (MIME) type of content. Content objects in the Content Integrator Java application programming interface

(API) are typically associated with a MIME type. The MIME type makes it easier for client applications to determine the viewing and authoring requirements for the native content. However, not all repositories provide an easy way to get the MIME type for their native content. If you cannot get the MIME type for the native content in your repository, your connector must make an effort to determine the MIME type on its own. Often, you can determine the MIME type by examining a file's extension. You can obtain the MIME type by utilizing the com.venetica.vbr.util.MimeTypeUtil.getMimeType(String filename) method.

### *Folder*

Next, you need to figure out how content is organized within your repository. Content Integrator uses a folder-based hierarchy for browsing the content of a repository. This question is essentially asking how Content Integrator represents this hierarchy for your repository. The hierarchy for your repository can simply be a view, allowing content to be filed in multiple locations. In other cases, the repository hierarchy can be representative of an underlying file system, with content being limited to a single location.

Content Integrator has a single, rooted folder hierarchy per repository. However, your repository might have multiple, orthogonal navigational hierarchies. In this case, you might need to decide how multiple repository hierarchies can be represented as a single hierarchy in Content Integrator. In this case, you can use an artificial root folder to contain the hierarchies and represent them as one hierarchy.

The repository can have more than one way to organize content, as illustrated in Figure 9-2 on page 453. For example, an item can have both a containing folder and a global category. If the category is more commonly used by the user to find content than the folder, and it is infeasible to incorporate both concepts into your folder hierarchy, you might want to map the category to the Content Integrator folder, instead of the more obvious mapping.

Finally, many repositories have no navigational organization at all and are accessed exclusively with search. In this case, your connector does not need to support folder functionality. Update the repository profile so that your connector correctly states that folders are not supported. We explain repository profiles later in this chapter.

Archived

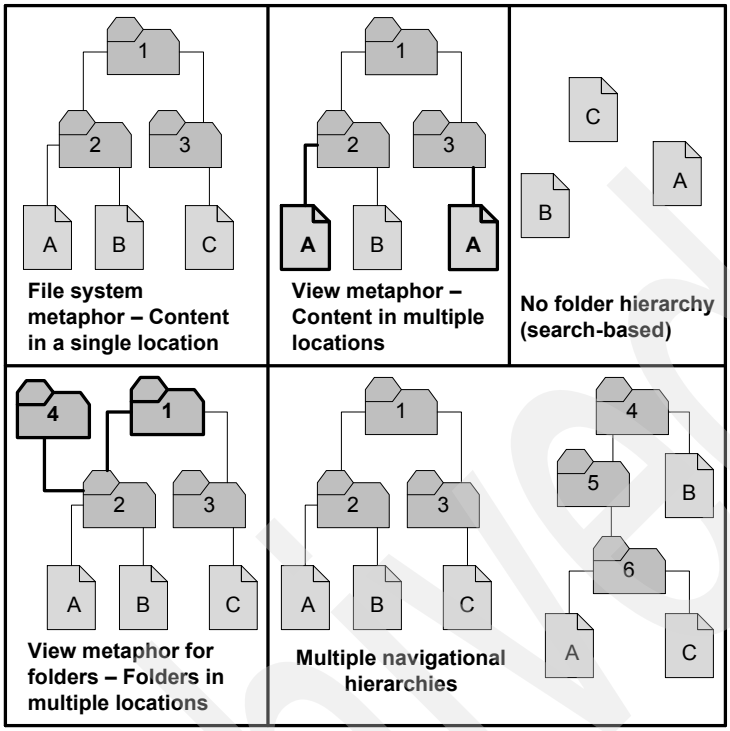



*Figure 9-2 Folder hierarchy examples*

Typically, a client uses the Java API class ItemFinder to browse repositories. If your connector supports browsing, it needs to execute ItemFinders appropriately. Executing ItemFinders properly is a relatively simple task for a standard, folder-based, hierarchical repository, such as your file system. When the ItemFinder requests a folder, such as `E:\` or `/mnt/data/content` directory, the File System Connector fetches the specified properties of all of the subfolders and files in the requested directory. If a logical and high-performance browsing metaphor exists for the repository, it is usually desirable to implement it in your connector.

### *Metadata*

If your repository stores metadata, it is important to understand how the data is formatted and what types of data, such as String, Integer, and DateTime, are stored. Item properties are always passed through the Java API as a String in the Property object, but defining the data type helps the client application to easily convert the value to the appropriate data type.

In the case of Date and DateTime types, the value that is stored in the Property object must be a long integer, representing the number of milliseconds that have

passed since January 1, 1970, 00:00:00 Greenwich mean time (GMT). The Time type is also a long integer, representing the number of milliseconds since midnight, local to the time zone in which the property was set. Java provides several easy ways to get a Date from the long integer using one of the classes in the Java.util.* and Java.text.* packages. More information exists about this topic in 8.2.6, “Locale and time zone support” on page 406.

You also need to determine if the metadata can be organized into classes and if those classes can be changed. Repositories can define a schema for a set of properties that are available on items of a certain type. This schema is known as an *item class* in Content Integrator. It can potentially be applied to any repository item, depending on how items are classified within the repository. The process by which item classes are defined can vary greatly. Certain repositories have immutable item classes that are set based on the item type; other repositories allow a repository administrator or user to define the classes that can be set on an item.

It is common for a certain set of properties to be defined across all items in a repository for a particular item type, such as a content item or a folder. In this case, a global item class can be defined for the connector, to serve as a way to contain those properties that apply to all item types.

#### *Unique identifier*

Items in a repository, such as content and folders, must have a unique identifier (ID) so that they can be routinely and consistently retrieved by the Java APIs. This identifier is meant to guarantee that the same content is always retrieved for any given unique ID. The com.venetica.vbr.client.RepoItemHandle class contains the unique item ID.

At times, the unique ID must be derived by your connector. For example, the file system connector uses a file’s absolute path to construct its unique identifier. However, in most cases, the repository will expose a unique ID as metadata. In this case, set the extended data type on the property description using IExtendedDataType.UNIQUE_ID.

### Designing around your repository’s API

Now that you have taken the time to understand how your repository is organized, you need to research what available programming interfaces it exposes. The most important point to consider is the functionality that is offered by each interface and the programming environment on which the functionality is available. Java APIs are usually preferred, because Content Integrator is a Java product, which makes connectors written in Java easier to implement. Connectors have been written to most of the common APIs in use, although a lack of a Java API is not a reason to stay away from connector development. Ultimately, the choice is based on which API provides sufficient functionality and

the comfort of the developer to program in that programming language. In rare cases, you can even use multiple APIs to access all of the desired functionality.

For cases where the expertise is with non-Java programming, you can expose a set of Web services, Component Object Model (COM) objects, or something similar that is tailored for use by Content Integrator. Your connector then acts merely as a pass-through to dramatically simplify the development of the connector.

Before you choose any particular approach, you must first have a good understanding of what is required of an Content Integrator connector. Connector developers need to remember several design points as they research the interfaces that are available to the native repository. We explain these design points in the following sections.

### *Removal of the application server requirement*

A recent improvement to Content Integrator, as previously noted in 6.7, “Recent product enhancements” on page 315, is the removal of the application server requirement. This improvement makes the deployment of an Content Integrator connector easier. The connector can run within the same Java virtual machine (JVM) as Content Integrator and the other connectors or it can be hosted in a separate JVM using the RMI proxy option.

### *Concurrency*

Your connector code does not need to be threadsafe, because the Content Integrator service provider interface (SPI) guarantees that your connector object instances only need to worry about one client (thread of execution) at a time. However, there can be multiple instances of your connector object that are all accessing your repository’s API at the same time from the same Java virtual machine (JVM). As a result, there can be a potential concurrency issue in your code at the class level, where static data is used, or in the repository’s APIs. Concurrency problems can have a big impact on your design, so they must be identified as early as possible. When you evaluate your repository API, confirm that it is threadsafe and contact the repository vendor for confirmation, if necessary. APIs that are not threadsafe require significant extra work on your part to coordinate access to them. And, you might also see a significant performance hit, because only one thread in each JVM is able to access your repository at a time.

Imagine a scenario in which there are 60 clients to your repository, resulting in 60 separate sessions; essentially, 60 possible separate concurrent threads. If 30 of these clients were trying to execute the same operation on the repository at the same time, 29 of them have to wait while one completes. We describe this scenario in Figure 9-3 on page 456. When you design your connector, you must

do everything in your power to avoid these kinds of resource contention situations.

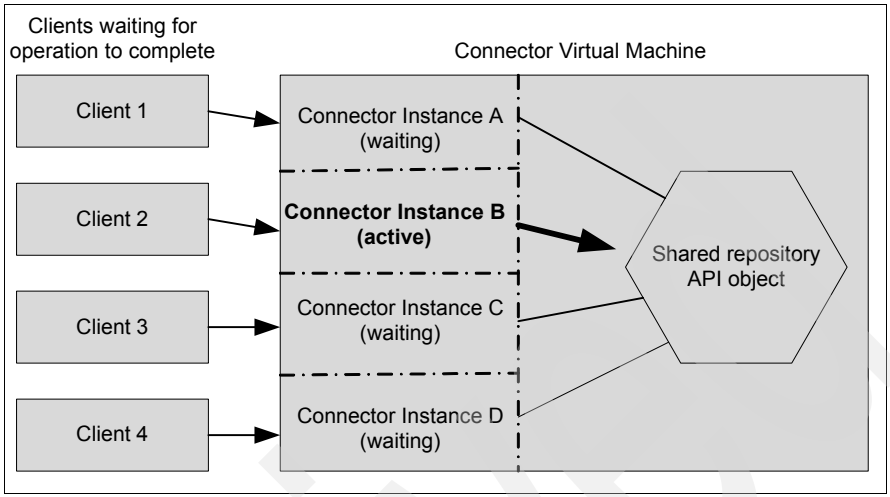



*Figure 9-3 Resource contention scenario*

### *Session management*

Session management can affect your connector design more than any other single factor. Let us look at how a client to Content Integrator expects a connector to behave so that we can fully understand how session management with your repository works. Client applications begin by initiating a connection with Content Integrator and requesting a Repository object. The Repository object represents one continuous session between the client and the connector. See Example 9-1.

*Example 9-1 Sample client application*

```
// Get connection to Content Integrator Access Services
User user = new User;
user.initialize();
// Request a repository session from Access Services.
Repository repo = user.getRepositoryByID( MyRepo );
// Logon to the repository.
repo.logon(myusername, mypassword, null);
```

Inside the *ConnectorBridge* class, the logon(AuthBundle) method is invoked. Example 9-2 on page 457 shows what this method might look like.

*Example 9-2 Sample logon method*

```
public void logon(AuthBundle bundle) throws LogonException {
   logDebug("In Logon");
   // Logon to repository according to its API
   repoLogon(_config, bundle); // Throws exception for bad creds
}
```

The client is authenticated with the repository through the connector, and a session is established from the perspective of the client. Each subsequent operation on the Repository object comes with the client's assumption that it is already logged on. With regular usage of the User or Repository object, the life cycle of the connector instance remains under the control of the Content Integrator client.

The connector must keep up the illusion of a session for the client, even for repositories that do not support one. It might be necessary to store the credentials and log on before each operation in the background for repositories that do not support sessions. For repositories that do support a continuous session, the connector logs in to the repository and saves the session.

**Store session in a non-static variable:** Be sure to store the session in a non-static variable. If the session object is stored in a static variable, and a user logs in as super user, everyone who logs in after that receives the same super user permissions. This scenario poses a potential security issue.

## Mapping your repository to the Java API: A case study

Mapping your repository to the Content Integrator Java API is the most important step in connector planning. The mappings that you choose now affect all aspects of your connector design. By this point in the process, we expect that you have a pretty thorough understanding of what your repository has to offer in terms of concepts, functionality, and interfaces. You might have started thinking about how to map certain repository concepts. Depending on your repository, these mappings can be fairly obvious. For certain repositories, a folder maps directly to an Content Integrator Folder, a defined group of properties to an item class, and so on. For other repositories, however, there can be several interpretations as to what a Folder can be.

### *File system case study*

Now, we look at the file system connector that ships with Content Integrator. While certain mappings are quite obvious, other mappings might not be, so it is important to think these mappings through to not waste any development time.

### *Content*

We start from the ground up. First, we consider what makes up our content, because the whole point of the connector is content retrieval and management. Next, we consider how other Content Integrator features, such as folders and item classes, can make that content and its metadata accessible. The obvious mapping to content is the files, which reside in the file system.

### *Folder*

Just as the content mapping is clear, the folder mapping is also clear. A standard file system consists of files and folders. The files are mapped as content, and the folders are mapped as folders, of course.

### *Metadata*

The mapping of metadata is a little less clear. We really only have access to system-provided properties, and there is no way to add additional properties to be stored, because a file system does not support the concept of a non-global item class. The file system connector, as provided with Content Integrator, does not support any properties on a file or a folder, only system attributes. Metadata can be introduced to a file system environment by incorporating a database server to store the metadata.

## 9.1.2 Implementing your new connector

After you have an understanding of how your repository maps to Content Integrator connectors, with a strong understanding of your repository's API, the rest of the connector development process is relatively easy. Next, we step through the process of building a brand new connector:

- Connector files: What are the required files to include
- Extending BaseBridge:
  - Content and folder retrieval
  - Browsing your connector: ItemFinder
  - Querying your connector: executeQuery()
  - Item creation
- Repository profile: BridgeFactory
- Configuration files

Archived

For most of the code examples, we use the file system connector as a base with modifications made where necessary in order to illustrate a point that might not apply to a file system. The source for the file system connector can be found within the Content Integrator installation at the following location:

*ICI_HOME*\docs\examples\java\spi\bridge

## Environment setup

When setting up your development environment to build your new Content Integrator connector, you must include certain files in your build path so that your classes compile. For Content Integrator, the only file that you need is the `vbr.jar` file. It contains all of the classes for the connector SPI that you use to define the functionality of your connector. You also need to include any files containing your repository's API in the build path to insure that your connector code is able to compile.

## Connector files

While the majority of the required development of your connector happens in one file, there are several files required in the creation of a new Content Integrator connector. The following list shows examples of the required files that we explain in this chapter.

- Java classes:
  - `yourpackage.ici.yourconnector.YourConnectorBridge`
  - `yourpackage.ici.yourconnector.YourConnectorBridgeFactory`
  - `yourpackage.ici.yourconnector.config.YourConnectorConfigBeanInfo`
  - `yourpackage.ici.yourconnector.config.YourConnectorConfigFactory`
- Supporting files:
  - `yourpackage/ici/yourconnector/config/yourconnector.properties`
  - `META-INF/MANIFEST.MF`

The functional part of your connector exists mostly in the *ConnectorBridge* class along with any helper classes that you create on your own. The additional classes do not require as much development time or planning as your *ConnectorBridge*. When setting up your development environment, you need to add *ICI_HOME*/lib/vbr.jar to your classpath for code compilation. Do not include it in your connector jar, though. Content Integrator puts the `vbr.jar` file on the classpath for you after your connector is deployed.

Archived

## Extending BaseBridge

Whether you are starting completely from the beginning or you have existing code that you want to integrate into your new connector, you need to begin by creating a class that extends the BaseBridge class from the Java SPI. We also recommend that the class implement the java.io.Serializable interface. See Figure 9-4. Classes, which implement the Serializable interface, also need to make sure that local variables are serializable, as well. There are other supporting classes, which need to be defined, but we discuss those classes later.

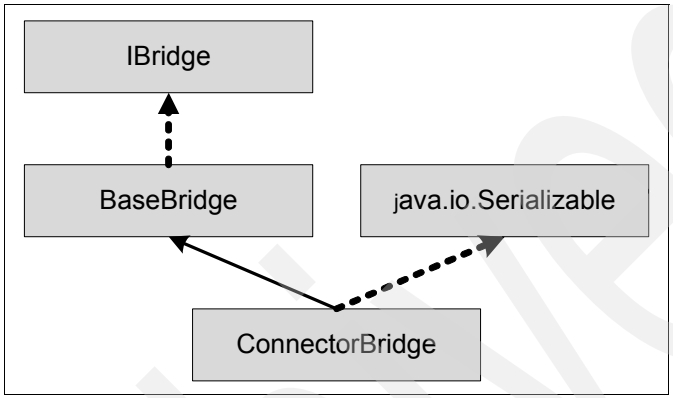



*Figure 9-4 Inheritance of ConnectorBridge*

Typically, your class that extends the BaseBridge class is named *ConnectorBridge*.java where *Connector* is the name of your connector: FileSystemConnector.java, for instance. BaseBridge takes care of the connector requirement of implementing IBridge. It is an abstract class, which contains all of the methods that you can conceivably implement in your connector. Many of these methods are not abstract and already have default behavior that is built in to save time. These methods either throw an UnsupportedActionException or provide other default behavior, such as returning empty arrays or null values. This approach makes it easier to add or remove functionality as necessary and ignore the functionality that does not apply.

### Logging on

The first methods that need to be implemented in your *ConnectorBridge*.java are the logon(AuthBundle) and logoff() functions. The logon and logoff methods are abstract members of BaseBridge. Your connector needs to maintain a session to your repository. You either need to store a session object to your repository, or if your repository does not support that, you need to store the credentials (AuthBundle) to authenticate future requests. AuthBundle can either be encrypted (sealed) or plain-text (unsealed). BaseBridge provides an unsealAuthBundle(AuthBundle) method, which can take sealed or unsealed

bundles. It simply ignores unsealed bundles. You can either pass all AuthBundle objects to that method or check AuthBundle.isSealed(AuthBundle) first.

The AuthBundle sealer relies on symmetric Blowfish keys. The same key must exist on both the connector side and the Content Integrator client side. The key file, `BlowfishKey.ser`, is located in *`ICI_HOME`*. With a standard connector deployment, which we describe later in this chapter, the key file is already available to the connector. The Content Integrator client also needs access to the same file, which is covered in Chapter 8, “Understanding the client APIs” on page 385.

Before unsealing the bundle, remember that you want to store sealed bundles, because unsealed bundles are vulnerable during serialization and memory dumps. When passing your bundle to unsealAuthBundle(), you can call AuthBundle.clone() to keep the original object sealed for storage. For repositories that do not support continuous sessions, you can store the credentials for later requests but still check the credentials in the logon() method to identify any problems early on when the user most likely expects them. Logon problems must result in a thrown LogonException.

The file system connector does not support authentication. The authorization comes from the account that is running the connector in the JVM. Example 9-3 shows how you might construct the logon method, but it does not really fit in the design of the file system connector.

*Example 9-3 The logon(AuthBundle) method*

```
public void logon(AuthBundle bundle) throws LogonException {
   logDebug("logon() - Logging on to repository");
   try {
      _bundle = bundle;
      if (bundle.isSealed()) {
         // Clone bundle so that original remains untouched.
         AuthBundle unsealed =(AuthBundle)bundle.clone();
         // unsealAuthBundle alters the object passed in.
         unsealAuthBundle(unsealed);
      }
   } catch (CloneNotSupportedException cnse) {
      throw new LogonException("Could not unseal bundle.");
   }
   try {
      // Make call to repository's authentication method via it's API
   } catch (Exception ex) {
      throw new LogonException("Invalid credentials.", ex);
   }
```

```
}
```

**Content Integrator connector logging:** The Java service provider interface's (SPI's) BaseBridge class contains several convenience methods for logging: logDebug, logError and logWarning. Using the Administration Tool, you can configure the connector to log each level or to not log each level.

Your repository can have multiple forms of authentication, so you might need to decide how to map the various credentials in your connector's logon method. The AuthBundle provides three fields that are transferred from the client to the connector: username, password, and optional. Most connectors do not require the optional parameter, but it can be useful if your repository requires that you qualify your credentials, such as with a Windows NT domain or a user role. You can also use the optional parameter if your repository supports something other than a user ID/password pair, such as a single sign-on (SSO) or biometric token.

We do not recommend that your connector use the optional parameter to control connector behavior. For example, it is not advisable for the optional parameter to determine the user's root folder in the repository. That model is detrimental to connector usability, because it requires that the user possess an unnecessary greater knowledge of repository internals to use the connector. The AuthBundle is meant to be used solely for authentication, and we recommend that all other configuration information is set with the Content Integrator Administration Tool.

## Repository items in Content Integrator

Because content retrieval and content management are the goals of any connector, item retrieval is at the top of the priority list. Before writing the connector code to map to items in your repository, it is important to understand what an item is to Content Integrator.

All items in Content Integrator are descendants of RepoItem, which provides generic functionality that applies to all items. Each RepoItem also contains a RepoItemHandle, which contains many things, including the ID of the item, the type of the item, the version of the item, and the repository where the item is held. RepoItem also contains the item's metadata in the form of Property objects, which contain a name and a string value. Many clients want to know more about the metadata, such as its datatype and whether it is queryable. These details are stored in a PropertyDescription object and attached to a Property object.

## Content and folder retrieval

Retrieving items (content and folders) in your connector consists of a few steps, which can be broken down into helper methods. It is important to understand each step and the purpose for that step in order to give the consuming client all of the data it is looking for. We summarize the steps:

1. Retrieve items from the repository.
2. Create a Content or Folder object to be returned.
3. Set item attributes on the Content or Folder object, including item class name.
4. Set item properties on the Content or Folder object, including system, global, and ad hoc properties.
5. Return the Content or Folder object to client.

### *Getting content or folder from the repository*

Item retrieval is implemented by overriding the `public RepoItem getRepoItem(RepoItemHandle)` method. As with all the other overridden methods of BaseBridge, we recommend that you always validate either that the session is still open or that the user credentials are still valid at the beginning of the method and throw a *LogonException* if they are no longer valid.

After you have verified that you can still access the repository, you need to determine which type of item is being requested. You can determine the item type in the RepoItemHandle.getItemType() method by comparing the return type with one of the constants in the IItemType interface. Many connectors might use a separate retrieval code based on the item type. It is most common for a connector to work with IItemType.CONTENT and IItemType.FOLDER, but other item types might apply, such as IItemType.WORK_ITEM or IItemType.WORK_QUEUE. If an item type is requested that is not supported, the connector throws an UnsupportedActionException.

After it has determined the item type, your connector needs to construct the item in the appropriate class and return it to the user. The four item classes currently in Content Integrator are Content, Folder, WorkQueue, and WorkItem. All of them extend RepoItem, so you can return any one of them from the getRepoItem method (Example 9-4).

*Example 9-4 The getRepoItem method*

```
public RepoItem getRepoItem(RepoItemHandle handle)
throws ItemNotFoundException, UnsupportedActionException,
VersionNotFoundException, PermissionDeniedException,
NotLoggedOnException {
   validateLogon();
   // Make sure the correct information is provided
```

```
   if (handle == null || handle.getItemID() == null) {
      throw new ItemNotFoundException("Can not retrieve items " +
            "with a null RepoItemHandle or ItemID");
   }
   String itemID = handle.getItemID();
   RepoItem item = null;
   // Check the item type and handle appropriately
   if (handle.getItemType() == IItemType.CONTENT)
      item = getContent(itemID, handle.getVersionID());
   else if (handle.getItemType() == IItemType.FOLDER)
      item = getFolder(itemID);
   else
      throw new UnsupportedActionException("Unsupported item type!");

   return item;
}
```

The FileSystemConnector might look like the getContent(String, String) method in Example 9-5. It has been altered from the actual connector source for simplicity. First, we check to make sure that the file actually exists (here we assume that the contentID is an accurate path). Next, we make sure that it is a file and not a directory. After we have confirmed that the file actually exists and that it is a file, we construct the Content object by calling a helper method.

*Example 9-5 Getting file system content*

```
public Content getContent(String contentID, String versionID)
throws NotLoggedOnException, PermissionDeniedException,
ItemNotFoundException, VersionNotFoundException {
   try {
      File file = new File(getAbsolutePath(contentID));
      // Make sure file exists
      if (!file.exists())
         throw new ItemNotFoundException("No file found with path: "
               + contentID);
      // Make sure file is not a directory
      if (file.isDirectory())
         throw new ItemNotFoundException("Content ID: " + contentID
               + " is not a valid file.");

      Content content = new Content();
      // Call method to set item attributes
      setItemAttributes(content, file);
      // Call method to set item properties
      setItemProperties(content);
```

```
      return content;
   } catch(RuntimeException ex) {
      String exceptionText = "RuntimeException in getContent(): "
         + ex.getMessage();
      logException(exceptionText);
      throw new ItemNotFoundException(exceptionText, ex);
   }
}
```

Similarly, the getFolder(String) method executes essentially the same commands as getContent() and looks like Example 9-6. We must first check if the root folder is requested and make a call to another helper method to get the correct root folder. Content Integrator uses the folder ID “-1” to represent a root folder, even for repositories that do not support a root folder. For repositories that support a root folder, this folder can be retrieved using either “-1” or the folder ID as assigned by the repository.

If another folder was requested, we need to make sure that it exists and that it is a directory. Next, we set any attributes and properties on it before returning it to the client (see Example 9-6).

*Example 9-6 The getFolder method*

```
public Folder getFolder(String folderID)
throws NotLoggedOnException, PermissionDeniedException,
UnsupportedActionException, ItemNotFoundException {
   try {
      Folder folder = null;
      if (folderID.equals("-1")) {
         return getRootFolder();
      } else {
         // Check if requested folder exists
         File dir = new File(getAbsolutePath(folderID));
         if (!dir.exists())
            throw new ItemNotFoundException("Folder ID: "+folderID+
                  " does not represent a path to valid directory.");
         // Check if requested folder is a directory
         if (!dir.isDirectory())
            throw new ItemNotFoundException("Folder ID: "+folderID+
                  " is a file not a directory.");

         folder = new Folder(folderID);
         // Call method to set item attributes
         setItemAttributes(folder, dir);
         // Call method to set item properties
```

```
            setItemProperties(folder);
         }
         return folder;
      } catch(RuntimeException ex) {
         String exceptionText = "RuntimeException in getFolder(): "
            + ex.getMessage();
         logException(exceptionText);
         throw new ItemNotFoundException(exceptionText, ex);
      }
}
```

### *Setting item attributes*

After you have retrieved the item or folder from your repository and verified its existence, you need to get the item attributes and apply them to your new Content or Folder item. Item attributes can be described as methods on the item, which describe it. For example, the RepoItemHandle, creation date, item class name, and anything else that was set on the actual Content or Folder object is an attribute. In Example 9-7, we use the FileSystemConnector as an example when gathering item attributes for a Content object. The method for Folder objects looks similar.

*Example 9-7 The setItemAttributes method*

```
protected void setItemAttributes(Content content, File file) {
   content.setName(file.getName());
   content.setHandle(new RepoItemHandle(_systemID,
         getItemIDFromPath(file.getAbsolutePath()),
         IContent.CONTENT));
   // Create MIME type based on extension
   content.setMimeType(_mimeTypeUtil.getMimeType(file.getName()));
   content.setPageCount(1);
   content.setDefaultFileName(file.getName());

   // Set permissions attributes
   content.setCanRead(file.canRead());
   content.setCanUpdate(file.canWrite());
   content.setCanDelete(file.canWrite());
   content.setCanChangeSecurity(false);
   content.setCanCheckout(false);
   content.setCanCancelCheckout(false);
}
```

### *Setting item properties*

Next, we set the item properties on the Content or Folder object. For the FileSystemConnector that we have been discussing, there really are not any properties, because there is no concept of an item class. As a result, we give a more generic example of how properties can be read and applied to the Content or Folder object. There are three kinds of properties:

- **System properties**

  System properties can apply globally to all items, such as revision dates or creator.

- **Properties with no property descriptions**

  Properties are the metadata of a repository's content, such as account number, phone number, or address information. They are a name/value pair.

- **Properties with property descriptions** (Example 9-8)

  Property descriptions are a description of the property, including the metadata about the property. A property description describes whether the property is writable, queryable, or selectable and its data type, among other information.

*Example 9-8 The setItemProperties method*

```
protected void setItemProperties(RepoItem item) {
   PropertyDescription pd = new PropertyDescription("PropDescr",
         IDataType.STRING, IDataType.EDITOR_TEXTBOX);
   pd.setIsQueryable(true);
   pd.setIsReadOnly(false);

   Property prop = new Property(pd);
   prop.setName("PropName");
   prop.setValue("PropValue");

   item.addProperty(prop);
}
```

The last step of getRepoItem() is to return the Content object to the client, which has been configured with all of the details of the item from the repository.

## Browsing your connector

Most connectors need to facilitate a quick property retrieval mechanism by executing ItemFinders appropriately. Quick property retrieval on a set of items is used in many Content Integrator client applications, such as those that provide a folder browsing metaphor for navigation.

Archived

You must implement the executeFinder(ItemFinder) method in BaseBridge to support ItemFinder execution in your connector. You need to provide a browse metaphor, if at all possible. Graphical browsing, if it can be supported, makes repository interaction much more intuitive for your users and, in general, provides a better user experience.

Figure 9-5 shows a RepoBrowser example that is included with Content Integrator.

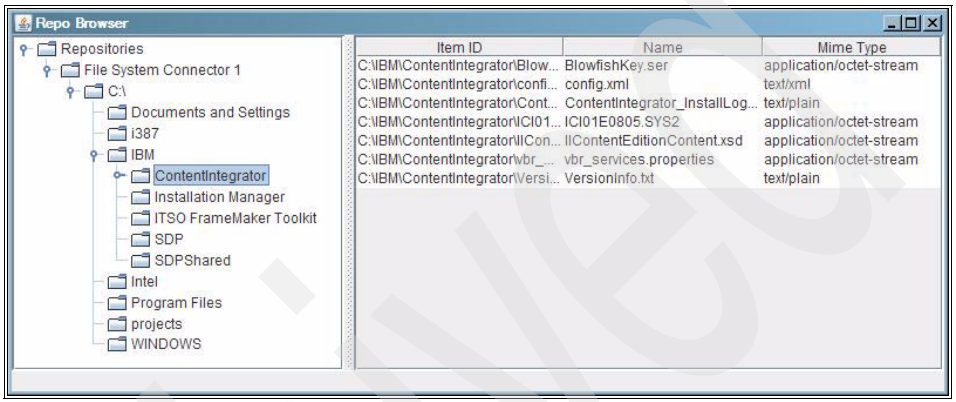



*Figure 9-5 RepoBrowser example that is included with Content Integrator*

ItemFinders can retrieve all of the items that are contained within a Folder or WorkQueue, as well as any additional item handles that have been requested. ItemFinders help to facilitate browsing. The RepoBrowser is a good example of a direct user interface to the repository, in which clicking a certain repository folder leads the user into that folder, much like a file system explorer does.

**RepoBrowser:** RepoBrowser is a Java Swing example that is included with Content Integrator and shows a good way to test direct browsing on a repository. You can obtain the source at this location:

`ICI_HOME/docs/example/java/swing/RepoBrowser.java`

RepoBrowser is already precompiled for you. There are two ways to run it:

- On Windows: Click **Start** → **All Programs** → **Content Integrator** → **Repository Browser Sample.**
- Run `cd ICI_HOME/bin`
  `run_sample swing.RepoBrowser`

### *Executing ItemFinders*

ItemFinder execution is not only for folders but also for a list of arbitrary repository items, because a stored list can contain any combination of disconnected items in your repository. ItemFinder execution is fairly simple. You call ItemFinder.getSearchContainer and ItemFinder.getSelectionCriteriaHandles to determine what the user wants your connector to retrieve from the repository. If selection properties have been specified, you also retrieve the specified properties of those items accordingly.

Your executeFinder(ItemFinder) returns a QueryResults object, which holds a number of ResultRow objects. Each result row holds a RepoItemHandle to identify the matching item and possibly a number of selection properties. Selection properties are item properties that the user explicitly requests to be returned with each result row. Selection properties make it possible for a user to browse through the repository while viewing certain properties on each item. Imagine a file system explorer on your computer — as you browse through the various folders and directories, the explorer shows you information about each file, such as the name or path, modification date, and the owner. Selection properties make browsing possible in Content Integrator, with any item properties that the user chooses. Example 9-9 shows the executeFinder() method, which handles files in a search container.

*Example 9-9 The executeFinder example only handles a search container*

```
public QueryResults executeFinder(ItemFinder finder)
throws ItemNotFoundException {
   validateLogon();
   QueryResults results = new QueryResults();
   RepoItemHandle hand = finder.getSearchContainer(0);

   String folderID;
   if (hand != null)
      folderID = hand.getItemID();
   else
      folderID = “-1”;

   File[] fileList = null;
   if (folderID.equals("-1")) {
      //Root folder
      fileList = new File[] {new File(_rootDir)};
   } else {
      //Non root folder
      File dir = new File(folderID);

      if (!dir.exists()) {
```

```
            String badID = "The specified search container "+folderID+
                  " does not exist.";
            logError(badID);
            throw new ItemNotFoundException (badID);
         }

         if (!dir.isDirectory()) {
            String notDir = "The specified folder ID "+folderID+
                  " is not a directory.";
            logError(notDir);
            throw new ItemNotFoundException (notDir);
         }

         // May return null if no permissions
         fileList = dir.listFiles();
      }

      if (fileList != null) {
         MimeTypeUtil util = new MimeTypeUtil();
         for (int i=0; i<fileList.length; i++) {
            RepoItemHandle handle = new RepoItemHandle(_systemID,
                  fileList[0].getAbsolutePath(), IItemType.CONTENT);
            ResultRow row = new ResultRow(handle, null);
            String mimeType = “”“;
            if (finder.getIncludeMimeTypes()) {
               mimeType = util.getMimeType(fileList[0].getName())
               row.setMimeType(mimeType);
            }
            if(finder.getIncludeItemNames())
               row.setItemName(fileList[0].getAbsolutePath());
            results.addResultRow(row);
         }
      }
      return results;
   }
```

In Example 9-9 on page 469, we only get the files that are directly in the search container (and we ignore any subfolders to reduce the complexity of the example). The sample file system connector is more complex and includes the handling of subfolders.

Archived

### *ItemFinder exceptions and maximum results*

There are a couple of other important considerations when you implement executeFinder for your connector. First, you might encounter an exception during the retrieval of a particular result. The exception might be a network issue, possibly a problem retrieving a particular item from the native repository, or anything at all. In this case, you add an exception to the QueryResults object, instead of adding a result. It is important that you catch this exception and add it to the QueryResults, and that you do not stop the execution of the ItemFinder. It is undesirable for an entire finder to fail because of one item.

Also, remember that ItemFinder supports a maximum number of returned results. Your executeFinder code needs to track the number of items that it has added to the result set and needs to cut off execution if that number reaches the maximum number. It is important to remember that exceptions do not count toward the maximum result limit, so make sure that you do not increment your counter on exceptions. Also, if you specify multiple search containers, the maximum result limit applies to the *entire* result set and not to each container.

## Querying in your connector

Adding query functionality to your connector means overriding the executeQuery(Query) method from BaseBridge. There are two types of queries that are supported by Content Integrator: property and full-text. However, you can combine them into one query if your connector supports combined queries.

Querying and item finder are similar. The executeQuery() method iterates through the various search options of the Query object and places it into the form that is required by your repository. The results that are received from the repository are subsequently parsed, and ResultRow objects are added to a QueryResults object.

One difference between Query and ItemFinder is that the results can obtain your results sorted according to a specified property. Your method needs to check if any sort specifications have been identified by calling Query.getSortSpecifications(). Each SortSpecification object contains a property name and a sort direction. If the property is not numeric, the results must be sorted lexically. In either case, where possible, we recommend that you allow the repository to do the sorting for you, because it is usually the fastest option. If your repository does not perform the sorting, you need to leave the sorting to Content Integrator, which sorts results automatically, by default. It is unnecessary to perform the sort in your Java code. If your repository provides sorting, you need to call Query.setUseNativeSort(boolean) with a true argument to tell Content Integrator not to sort automatically.

Query, unlike ItemFinder, also requires relatively simple parsing to interpret query expressions correctly. Full-text query expressions are retrieved with

Query.getFullTextQueryExpression(), and property query expressions use Query.getQueryExpression(). If both types of query exist on your Query object, your connector executes a combined query. Combined queries must only apply if your connector can execute the combined query as a single operation. We do not recommend that your connector attempt to execute both types of query and programmatically combine the result sets. In that case, it is better to not support combined queries and to document that in the repository profile accordingly by returning false from getCanQueryPropertiesAndFullText().

Additionally, Query also includes the option to limit searches to these objects:

- A container, such as a folder or a work queue
- An item class

If a search container is specified, and your connector supports search containers, the results that are returned must exist within that container. If your connector does not support search containers, but a search container is specified, you must throw an exception to inform the client of the problem. The same situation applies if an item class is specified.

## Query expressions

There are two methods on the Query object, which return QueryExpression objects: getFullTextQueryExpression() and getQueryExpression(). A QueryExpression object specifies the seach criteria expression that is used when querying a repository.

This example shows the sample query expression content:

```
author=‘John Smith’ AND category=‘ICI’ AND NOT rating<‘3’
```

Each QueryExpression is composed of any number of QueryExpressionElement objects. A QueryExpressionElement can exist in three forms:

- An expression in the form [property][operator][value] for example: category=‘ICI’. See Table 9-1 on page 473 for a list of valid operators.
- A conjunction. These conjunctions are valid:

  ```
  'AND'
  'AND NOT'
  'OR'
  'OR NOT'
  ```

- Opening or closing parentheses: ‘(‘, ’)’

*Table 9-1 Valid expression operators*

| Operators | Description |
|---|---|
| = | Equals |
| <> | Not equals |
| > | Greater than |
| >= | Greater than or equal |
| < | Less than |
| <= | Less than or equal |
| LIKE | Is like |
| ISNULL | Is null |
| ISNOTNULL | Is not null |
| NOTLIKE | Is not like |
| BETWEEN | Is between |
| NOTBETWEEN | Is not between |
| IN | Is in |
| NOTIN | Is not in |

Your connector needs to parse through each QueryExpressionElement and use the objects to create a corresponding query to your repository.

### *Interpreting query expressions*

While there are many ways to construct your algorithm to form repository queries, remember that both full-text queries and property queries are composed of QueryExpressionElement objects but the difference is in the *property*='*value*' type of expression. In a property query, the *property* portion of the expression has an obvious, literal meaning while in a full-text query, the *property* portion is limited to either a *word* or *phrase,* as shown in Example 9-10.

*Example 9-10 Syntax of a full-text query expression*

```
WORD="football college sports"
PHRASE="College football season"
```

These portions of the expression need to be interpreted correctly by your connector. If something else is in the *property* portion, an exception must be

thrown. In Example 9-10 on page 473, the WORD query is split up by your connector as though three separate query expressions were specified.

It is also important to notice that each *value* term is surrounded by single quotation marks. The requirements of your repository might conflict with the use of single quotation marks so they might need to be removed. There are two static methods to assist in this area: QueryExpression.quote(String) and QueryExpression.unquote(String).

Depending on the requirements of your repository, you might also need to pay attention to double negatives in the expressions. If your repository does not support double negatives, you need to check for that condition and correct it for each QueryExpression.

### *Invalid literals*

Invalid literals can be a frustrating complication in query expressions. Your repository might have certain characters or strings, which are reserved, have special meaning to the repository, and require special handling in your connector. For example, if a repository allows for the string literal 'link:' to quickly search for a link, you must handle the case where that string is passed in not as the reserved word but as an actual value.

In many cases, the repository offers an escape character for these circumstances to avoid conflict with reserved words. If your repository provides an escape character or sequence, we recommend that you always use it. If there is any literal that cannot be escaped for any reason, add it to the list of invalid literals in the repository profile, which we discuss later in this chapter. Your connector needs to check for invalid literals and throw an exception whenever they are encountered. Deciding to merely pass these invalid literals along to the repository is a bad idea. An unknowing user might use an invalid literal and the results they receive back might not be what they expect. Throwing an exception when these invalid literals are used helps to provide consistent results.

### *Wildcards*

The query functionality of your connector must also be able to handle wildcards. The wildcards are the asterisk (*) for multiple characters and the question mark (?) for single characters, according to the Content Integrator query language. The wildcard characters can only be used with the LIKE operator in either full-text or property searches. If your repository does not support wildcard searching, throw an exception for queries, including wildcards. The two wildcard characters, when used with other operators, are considered simply literals and are passed along as literals.

Check with your repository, however, to see if the wildcard characters must still be interpreted as literals, even if the LIKE operator was not used. In this case,

you might need to escape the characters so that they can be interpreted as literals as intended.

## Item creation

To add support to your connector for item creation, you need to implement two methods from the BaseBridge class:

```
public String createItem(IContent item, String folderID)
public String createItemFromFiles(IContent item, String folderID)
```

Because the first parameter of the two methods is an IContent object, these methods only allow for the creation of content and folders. The folderID parameter is the containing folder in which to create the item.

The difference between these two methods is in the way that they attempt to upload the native content. The createItem(…) method expects that the client passed the native content to the connector using a byte array. The createItemFromFiles(…) method expects that the client passed the native content to the connector using file streaming. We always recommend that you use the streaming option, whenever possible. The byte array method loads the entire native content into memory all at one time. And, for extremely large files, the available memory can fill up quite quickly and potentially result in out of memory errors. The streaming method, however, passes the native content in broken up chunks, reducing the amount of consumed memory in the Java virtual machine.

In the file system connector, we override the two methods, as shown in Example 9-11, and pass the request to a helper method and include a boolean parameter to specify the way to handle the native content upload. It is up to the client to call the appropriate method.

*Example 9-11 Overridden create item methods*

```
public String createItem(IContent newItem, String containingFolderID)
throws NotLoggedOnException, PermissionDeniedException,
ItemCreationException {
   validateLogon();
   // Pass false to specify that we are not using the streaming method
   return createItem(newItem, containingFolderID, false);
}

public String createItemFromFiles(IContent newItem, String
containingFolderID) throws NotLoggedOnException,
PermissionDeniedException, ItemCreationException {
   validateLogon();
   // Pass true to specify that we are using the streaming method
```

```
   return createItem(newItem, containingFolderID, true);
}
```

In our helper method, the first step is to check the containingFolderID that is passed in to verify that it is a valid directory in which to create content. As with the previous examples, we assume that the folder ID is the correct path to the directory. The actual connector needs to verify and possibly to correct the path for the ID first. See Example 9-12.

*Example 9-12 Verify that the containing folder exists and allows writes*

```
protected String createItem(IContent newItem, String
      containingFolderID,boolean useFiles)
throws NotLoggedOnException, PermissionDeniedException,
UnsupportedActionException, ItemCreationException {
   try {
      String newContentID = null;
      File parentDir = null;

      try {
         // Get File object
         parentDir = new File(getAbsolutePath(containingFolderID));
      } catch(ItemNotFoundException ex) {
         throw new ItemCreationException("Containing folder "
               + containingFolderID + " does not exist.");
      }
      // Check if it is a directory and can be written to
      if (!parentDir.isDirectory() || !parentDir.canWrite())
         throw new ItemCreationException("Containing folder "
               + containingFolderID + " does not exist or is" +
                     " not a writable directory.");
```

The second step in Example 9-13 is to determine whether we are creating a new file or a new folder and to create the appropriate item.

*Example 9-13 Create a new file or a new folder*

```
      if (newItem instanceof Content) {
         Content content = (Content) newItem;

         if (content.getIsMultiPart())
            throw new UnsupportedActionException("Failure creating new"
                  + " content. Multi-part content is not supported.");

         File newFile = new File(parentDir.getPath() + File.separator
```

```
                + new File(content.getDefaultFileName()).getName());

            newContentID = getItemIDFromPath(newFile.getAbsolutePath());

            if (newFile.exists())
               throw new ItemCreationException("Cannot create new file "
                     + newContentID + ". The file already exists.");

            if(useFiles) {
               // Use file streaming method
               File inFile = content.getNativeContentUploadFile();
               FileUtil.copyFile(inFile, newFile);
            } else {
               // Use byte array method
               byte[] buff = content.getNativeContentUploadBuffer();
               OutputStream out = new FileOutputStream(newFile);
               out.write(buff);
               out.close();
            }
         } else if (newItem instanceof Folder) {
            Folder folder = (Folder) newItem;
            File newDir = new File(parentDir.getAbsolutePath()
                  + File.separator+folder.getName());
            newDir.mkdir();
            newContentID = getItemIDFromPath(newDir.getAbsolutePath());
         }
         return newContentID;
      } catch(IOException ex) {
         String exceptionText = "IOException in createItem(): "
            + ex.getMessage();
         logException(exceptionText);
         throw new ItemCreationException(exceptionText, ex);
      } catch(RuntimeException ex) {
         String exceptionText = "RuntimeException in createItem(): "
               + ex.getMessage();
         logException(exceptionText);
         throw new ItemCreationException(exceptionText, ex);
      }
}
```

After you have verified all of the parameters, you might need to loop through any attributes and properties that need to be stored in the repository. Depending on your repository's API, these attributes and properties might need to be added to the item before creation or after creation. Many system attributes can be set

automatically, such as creation date and creation user, depending on your repository.

After handling the metadata, we check the boolean parameter passed in to determine whether we use the byte array method or the file streaming method. Following the appropriate method, the native content from the Content object is passed to the repository. For the byte array method, the native content is passed with the use of the Content.getNativeContentUploadBuffer(), which returns a byte array. Be careful not to confuse that method with the Content.getNativeContent() method, which also returns a byte array but is intended for use by the client.

After your content has been created on the repository, your createItem(…) method needs to return a String item ID for the item. In many cases, the repository API itself returns a new item ID, which can be passed along to the client. In the case of the file system connector, the ID is based on the absolute path of the file/folder, so it is the connector's responsibility to grab the ID and, if necessary, alter it before returning it to the client.

Remember that the item ID returned must always point back to the same exact item in the repository every time. Guaranteeing item ID uniqueness insures that the user always gets the exact content that the user expects for any query. The same principle applies to version ID if your repository supports versions.

## Factory class

The next required class to include is the ConnectoryBridgeFactory class, which extends the BridgeFactory class. This class is used by Content Integrator to create instances of your connector, which are handled by the createBridge() method. See Example 9-14.

*Example 9-14 The createBridge method that is defined in FileSystemBridgeFactory*

```
// bridgeInstanceID A number used to identify the connector in the log.
// systemID The unique identifier of the repository being accessed by
// the connector instance
// config The configuration object, unique subclass of BridgeConfig for
// each connector type, needed to provide the data to configure the
// connector.
public IBridge createBridge(long bridgeInstanceID, String systemID,
   BridgeConfig config)
throws BridgeCreationException {
   _systemID = systemID;
   _description = config.getDescription();

   return new FileSystemBridge(getSystemID(), bridgeInstanceID, config,
         getRepositoryProfile());
```

```
}
```

Next, we review the next part of the bridge factory class, which is the methods that help define the repository profile for your connector. The repository profile is used by the client to determine what is supported by your connector, which makes for better error handling and feature determination.

For a list of available methods to override for your repository profile, review the SPI Java docs for the BridgeFactory class. A few examples of these methods are getCanCreateContent(), getCanCreateFolders(), and getCanQuery(). Several of these methods are given a default value in the definition of BridgeFactory, but many of the methods are abstract, so you need to define them yourself. Override the methods as necessary to add or remove support for that particular functionality in your connector. Many of these methods return boolean values, but other methods return information, such as the maximum number of query results and the invalid literals for each kind of repository query. See Example 9-15.

*Example 9-15 Overriding getCanCreateContent in ConnectorBridgeFactory*

```
public boolean getCanCreateContent() {
   return true;
}
```

Note that although the method in Example 9-15 from the repository profile shows that content can be created, it is only a statement about whether the connector supports content creation. It does not make any guarantees about user permissions, which can prevent the creation of content for many reasons. If permissions do not allow the creation of content, check for that situation when the attempt to create content is made in *ConnectorBridge*.createItem(). Throw an exception, in that case, to let the client know that the client does not have sufficient privileges to perform that action.

## Configuration files

There are a few configuration files, which were mentioned in the beginning of the Connector files section. Creating these configuration files and defining these configuration files are quick and easy tasks.

### *ConnectorConfigBeanInfo*

*`ConnectorBridge`* has access to a BridgeConfig, which contains all of the configuration information about your connector. This information is configured using the Administration Tool, which scans the *`ICI_HOME`*`/ejb` directory for all of the connector jars when it loads. Next, the Administration Tool uses the *`ConnectorConfigBeanInfo`* (which extends BridgeConfigBeanInfo) to determine

the properties that are required by the connector. When a user configures a connector in the Administration Tool, the user is presented with many configuration fields, including standard fields and any properties that are specific to this connector read from the *ConnectorConfigBeanInfo* class. Example 9-16 shows how custom properties are added to a connector.

**Administration Tool:** The Content Integrator Administration Tool is used to configure any connectors in your environment, along with data maps and session pools. You run this tool by using a batch (Windows) file, which can be found in *ICI_HOME*/bin/Admin.bat file, or by using a script (UNIX) file, which can be found in the *ICI_HOME*/bin/Admin.sh file.

*Example 9-16 Adding a custom property to your connector*

```
public PropertyDescriptor[] getPropertyDescriptors() {
   try {
      // Get the property descriptor list from base BeanInfo.
      PropertyDescriptor tv[] = super.getPropertyDescriptors();
      Vector<PropertyDescriptor> pv = new Vector<PropertyDescriptor>();
      int index = 5;

      // Custom property to specify the root directory
      // We use index to specify where in the order of properties
      // to put the new property
      PropertyDescriptor rootDirectory =
         new HashedPropertyDescriptor("rootDirectory",
               _beanClass, ++index);
      rootDirectory.setDisplayName("Root Directory");

      // Fill vector with existing properties
      for (int i = 0; i < tv.length; i++)
         pv.addElement(tv[i]);

      // Add new property to vector
      pv.addElement(rootDirectory);
      PropertyDescriptor rv[] = new PropertyDescriptor[pv.size()];
      pv.copyInto(rv);

      return rv;
   } catch (IntrospectionException e) {
      e.printStackTrace();
      return super.getPropertyDescriptors();
   }
}
```

The custom property can then be read in your connector by accessing the BridgeConfig object that is passed into the bridge constructor. Example 9-17 shows how to read the *rootDirectory* custom property that was specified in Example 9-16 on page 480.

*Example 9-17 Reading the custom property*

```
public static final String ROOT_DIR = "rootDirectory";

protected FileSystemBridge(String systemID, long bridgeInstanceID,
      BridgeConfig config, RepositoryProfile profile) {
   _systemID = systemID;
   _bridgeInstanceID = bridgeInstanceID;
   _config = config;
   loadMimeTypes();

   // Read rootDirectory property from bridge config
   String rootDir = getConfig().getPropertyString(ROOT_DIR);
   if (rootDir != null && rootDir.length() > 0
         && new File(rootDir).exists()) {
      _rootDir = rootDir;
   }
}
```

### *ConnectorConfigFactory*

The *ConnectorConfigFactory* is also a relatively simple class to put together. It describes how the connector configuration is created and accessed. It holds the factory classes and connector properties files for your connector.

### *The connector.properties text file*

The final configuration file is the connector.properties text file. It provides a default configuration for your connector. Even if a few of the properties must be configured by the user, we recommend that you provide a default value for the properties. Default values can be helpful by providing an example of the proper format or by showing the context to assist the user in determining how to configure the property. The `connector.properties` file is automatically loaded from the same package as the *ConnectorConfigFactory* (for example, *yourpackage.ici.yourconnector.config*), so it is important that they are kept together. See Figure 9-6 on page 482.

Archived

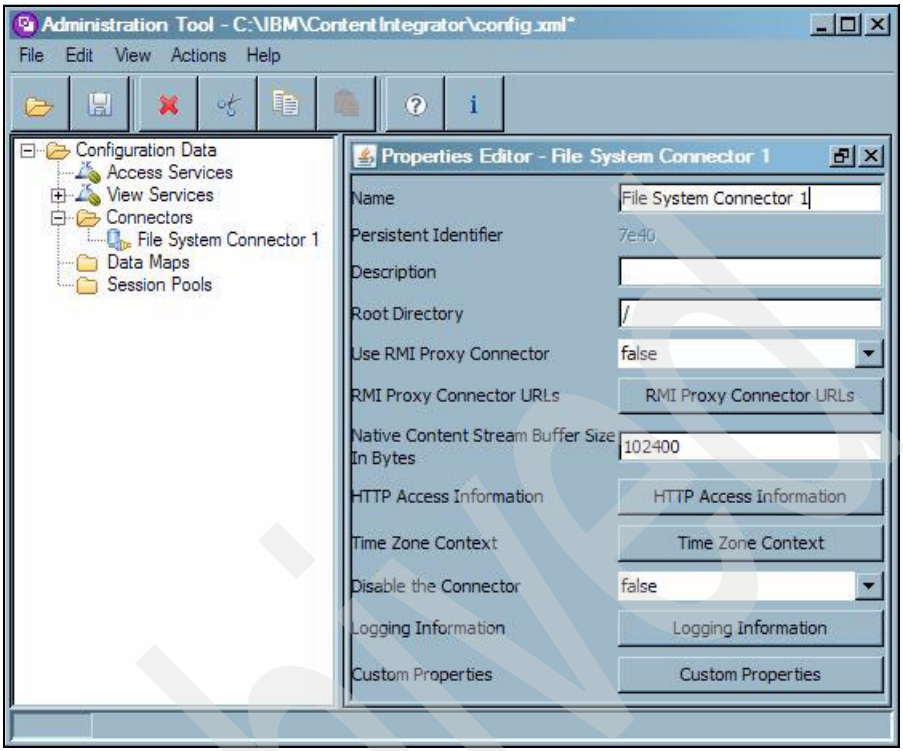



*Figure 9-6 Using Content Integrator Administration Tool to configure connector*

## Files and packages

Earlier in this chapter, we provided a list of files and their packages. It is important to note that this structure is merely a suggestion. The files can be included in a variety of ways. Content Integrator relies on a properly configured `MANIFEST.MF` file, as shown in Example 9-18, to direct it to the location of the *`ConnectorConfigBridgeFactory`* class in the jar file. With the location of this class, Content Integrator is able to configure and create your connector instance.

*Example 9-18 Sample MANIFEST.MF file taken from file system connector jar*

```
Manifest-Version: 1.0
Ant-Version: Apache Ant 1.6.5
Created-By: IBM Corporation
Copyright-Notice: Licensed Materials - Property of IBM 5724-J31 (c) Co
 pyright IBM Corp. 2000, 2007, 2008 All Rights Reserved. US Government
  Users Restricted Rights - Use, duplication or disclosure restricted
 by GSA ADP Schedule Contract with IBM Corp.
Class-Path: lib/vbr.jar
```

```
Name: com/venetica/vbr
Specification-Title: Content Integrator
Specification-Version: 8.5
Specification-Vendor: IBM Corporation
Implementation-Title: com.venetica.vbr
Implementation-Version: 200811071638
Implementation-Vendor: IBM Corporation

Name: com.venetica.vbr.ejb.bridge.filesystem.config.FileSystemConfigFa
 ctory
ObjectFactory: true
```

You can extract the contents of any connector in the *ICI_HOME*/ejb directory to look at its MANIFEST.MF file, as an example. Most integrated development environments (IDEs) or build scripts, such as Apache Ant, construct this file for you. Essentially, your connector jar file contains the folder structure of your connector classes and a folder called META-INF, which contains the MANIFEST.MF file. The important entries of your MANIFEST.MF file are *Name* and *ObjectFactory,* as displayed in Example 9-18 on page 482. *ObjectFactory* must be set to true.

## 9.1.3 Building, deploying, and configuring your new connector

When you want to test your connector, you first need to create a jar file containing all the necessary files. This task is quite simple. This section takes you through the following steps:

- Building your connector jar file
- Deploying your new connector
- Configuring your new connector

### Building your connector jar file

Building your connector jar file is simple. Your new jar file contains the folder structure of your packages, your classes, and the supporting files. You new jar file also contains the META-INF folder containing the MANIFEST.MF file, which points to your *ConnectorConfigFactory* class. Many IDEs provide a method for exporting your jar file for you, including the creation of the manifest file. You can also create it manually, but we recommend that you use the available tools to avoid human errors.

Do not include the jar files, which contain the API class files that expose access to your repository, in your connector. We address these files in the next section.

Archived

## Deploying your new connector

After you have built the jar file containing your Java classes and supporting files, including the MANIFEST file, you must deploy your connector jar file. Connectors might need to be deployed in two locations, depending on your configuration. The installation of Content Integrator where you launch the Administration Tool needs to have it in its class path. The Administration Tool relies on a classloader to read all of the jar files at the *ICI_HOME*/ejb directory, which is where you must deploy your jar file. If your environment uses the Remote Method Invocation (RMI) Proxy Connector, the connector jar must also be deployed to the same location on that machine.

You also need to include your repository's API jars in the classpath so that your connector is able to access your repository successfully. One way to add them to the classpath is to copy them to the *ICI_HOME*/lib directory where your connector is running. In the case where RMI is used, the repository API files only need to be deployed to that machine. The other way to include your repository's API jars in the classpath is to edit the RMIBridge.bat file on Windows or the RMIBridge.sh file on Linux/UNIX.

## Configuring your new connector

After the new connector is deployed and the Administration Tool is loaded, you add your connector to the configuration by right-clicking the **Connectors** node of the tree and selecting your connector from the list of available connectors. An entry is added as a child element to the tree beneath Connectors and displays a configuration window to the right. There, you can enter all of the necessary information and save it. Now, you can test the connection by right-clicking your connector in the tree at the left and choosing **Test Connection**. Enter the proper credentials for your repository, and you get a confirmation window returned. See Figure 9-7.

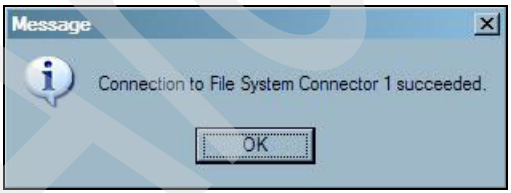



*Figure 9-7 Successful connection to file system*

At this point, you have a newly created connector, which has been deployed to your environment and configured to allow clients to access it and connect to your repository.

## 9.2 Extending the existing Content Integrator connector

If you can determine that Content Integrator already provides a connector for your repository but you are interested in adding or removing functionality, you can develop a connector, which extends the existing connector. This method allows you to make use of existing and tested functionality without having to write the code yourself. We cover the following topics in this section:

- Planning and design
- Implementing
- Building, deploying, and configuring your connector

As we step through these features in extending a connector, be aware that we assume that you are familiar with the creation of a new connector by reading the first section of this chapter.

### 9.2.1 Planning and design

The planning and design that goes into extending a connector can be similar to writing a new connector as we described in 9.1, “Implementing a Content Integrator connector” on page 450. Depending on the functionality changes that you intend to make, it can be a quick development experience or a long one.

#### Choosing the existing connector

In order to determine if you can extend the connector, you must first confirm that there is in fact a connector that shipped with Content Integrator that supports your repository, especially the version of your repository. Consult this support document to determine which connector, if any, supports your repository:

http://www.ibm.com/support/docview.wss?rs=86&uid=swg27015930

If you are still unsure, you might be able to contact the Content Integrator support team to confirm support.

After you have determined that Content Integrator provides a connector that supports your repository, and you are familiar with your repository’s API, you need to decide what functionality you want to replace, remove, or add.

### 9.2.2 Implementing

The steps to extend a connector are similar to those steps to develop a new connector.

Archived

## Environment setup

When setting up your development environment to build your new Content Integrator extended connector, you must include a few files in your build path so that your classes compile. For Content Integrator, you must include these files:

- The `vbr.jar` file. This file contains all of the classes for the connector SPI that you use to define the functionality of your connector.

  *ICI_HOME*/lib/vbr.jar

- The connector jar file to extend. Your connector must be aware of the classes and available methods from the connector that you are extending.

  *ICI_HOME*/ejb

If your connector makes calls using your repository's API, you need to include any files containing your repository's API in the build path. With these files, your connector code is able to compile those calls to that API that you write into your connector.

## Connector files

The list of files required for a connector, which extends an existing connector, is the same list as for a brand new connector. The difference is that your classes extend the matching class from the connector rather than directly extending a class from the Content Integrator SPI.

### *Java classes*

Include these Java classes:

```
yourpackage.ici.yourconnector.YourConnectorBridge
yourpackage.ici.yourconnector.YourConnectorBridgeFactory
yourpackage.ici.yourconnector.config.YourConnectorConfigBeanInfo
yourpackage.ici.yourconnector.config.YourConnectorConfigFactory
```

### *Supporting files*

Include this supporting file:

```
yourpackage/ici/yourconnector/config/yourconnector.properties
```

## Determining which methods to override

All of the listed classes are required, but where you actually make functionality changes depends on what you need from this new connector. If you need to remove or change specific functionality, you need to understand where that functionality comes from.

One quick step that is required in adding or removing functionality is to make this change in the repository profile that we discussed earlier in this chapter. The

repository profile is defined in the *ConnectorBridgeFactory* class. You can see the list of repository profile methods in com.venetica.vbr.ejb.spi.BridgeFactory in the SPI documents. Most of these methods return a boolean value that states whether that functionality is supported.

It might be necessary to write a sample client application to make calls to the methods that you want to use to determine the value they return. For example, if the existing connector allows you to create content but does not allow you to create folders in your repository, you might want to add support for folder creation. And perhaps, you want to remove the ability to create content to be more restrictive. You need to override two methods of the repository profile in *ConnectorBridgeFactory* to correctly report the desired functionality changes. See Example 9-19.

*Example 9-19 Overriding getCanCreateFolders() to add support for folder creation*

```
public boolean getCanCreateFolders() {return true;}

public boolean getCanCreateContent() {return false;}
```

Updating these methods alone does not change your connector's actual functionality, however. It is a good practice for a client to Content Integrator to check the repository profile to make sure that an action is supported prior to attempting to execute that action. In any case, you need to override any methods that provide the actual functionality.

## Where to add new functionality

The majority of functionality begins in the *ConnectorBridge* class. Following Example 9-19 for adding support for folder creation and removing support for content creation, the method to be overridden in your *ConnectorBridge* class is the createItem(IContent item, String containingFolderID) method. The createItem method is responsible for creating both content and folders. Therefore, you must know which kind of item creation the existing connector supports before you begin to write your new method. You might want to call the parent method to handle the other types of items that it already creates. See Example 9-20.

*Example 9-20 Overriding createItem()*

```
public String createItem(IContent item, String containingFolderID)
throws RemoteException, VeniceBridgeException {
   if (item instanceof Folder) {
      // Create folder here
   } else if (item instanceof Content) {
      // Throw an exception here to notify the client
      // The client should have checked the repository profile first
      // though
```

```
        throw new unsupportedActionException("Content creation is not "+
              "supported by this connector.");
    } else {
        // There should not be any other kinds of items to create but
        // we rely on the parent method to handle this.
        return super.createItem(item, containingFolderID);
    }
}
```

In Example 9-20 on page 487, we first check which type of item is passed in. If it is a folder, we make the necessary calls to our repository's API to create a new folder. If it is content, we throw an exception to inform the client that content creation is not supported. If something other than a Folder or Content object was passed in, we rely on the createItem() method of the existing connector to handle it. If it is an unsupported item type, the inherited createItem() throws the exception for us. We want to minimize our impact and rely on the parent connector as much as possible.

Make sure that you have error and exception handling in any block of code in which you are adding or removing functionality. For example, although the repository profile now states that folders can be created, that is only a statement about the connector's support of that functionality. If the calling user does not have sufficient privileges to create folders, your new block of code in createItem() checks for those privileges and throws an exception to let the client know that the client does not have the permissions to perform that action.

### 9.2.3 Building, deploying, and configuring your new connector

The process to build, deploy, and configure your connector is the same process as the steps for a new connector. Refer to 9.1.3, "Building, deploying, and configuring your new connector" on page 483 for more details. Do not forget to test the connection as the last step.

## 9.3 Connector plug-ins

Connector plug-ins allow you to create multiple custom methods that can be easily executed from Content Integrator client applications. In this section, we cover the following topics:

- The purpose of a connector plug-in
- Creating a connector plug-in
- Packaging a connector plug-in
- Configuring a connector plug-in

## 9.3.1 The purpose of a connector plug-in

The Java SPI provides methods in connectors, which give Content Integrator clients exposure to many ways to manage the content in back-end repositories. In certain cases, a desired feature might not be exposed by the Java SPI, which is where the connector plug-in fits. Figure 9-8 shows the location of the connector plug-ins in Content Integrator.

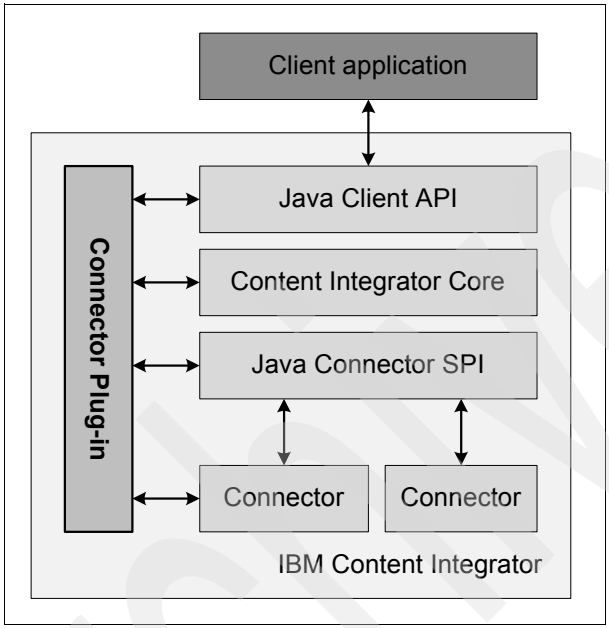



*Figure 9-8 Location of plug-ins in Content Integrator*

For example, through the standard Content Integrator connectors, currently you cannot create, delete, or modify an item class in the back-end repository. If your repository's API exposes this functionality and you want to control that functionality from an Content Integrator client, you create a connector plug-in.

## 9.3.2 Creating a connector plug-in

The steps in creating a connector plug-in are similar to regular connector development. We describe these steps:

- Planning the connector plug-in
- Implementing the connector plug-in

A quick overview of the connector plug-in feature is available in the information center:

Archived

http://publib.boulder.ibm.com/infocenter/ce/v8r5/topic/com.ibm.discovery.ci.conndev.doc/cdncd005.htm

## Planning the connector plug-in

Assuming that you have determined that there is a connector that works with your repository, you first evaluate whether the functionality you want to add can be accomplished by extending the connector instead. We recommend that you add functionality through extending the connector wherever possible. If there are no available methods in the connector SPI, creating a connector plug-in might be the answer.

Connector plug-ins are essentially limitless in their capabilities. Choose connector plug-ins for functionality that a connector does not have. You might decide that you need to be able to update a single page in a multipage item, but it is not supported by Content Integrator connectors. If your repository's API supports updating single pages in a multipage item, you can choose to build a connector plug-in to meet this need. For a list of functions that are not in Content Integrator, see 8.2.8, "Non-supported operations" on page 409.

A sample connector plug-in is available with Content Integrator called *BaseConnectorPlugin*. We follow that sample to explain how to build a connector plug-in. You can locate the sample connector plug-in at this location:

*ICI_HOME*\docs\examples\java\com

## Implementing the connector plug-in

In order to write a connector plug-in, you need to include the `vbr.jar` file in your development environment's classpath. It contains all of the necessary classes that your connector plug-in must implement. If your connector plug-in makes calls to your repository's API, you must also include the jar files containing your repository's API.

Connector plug-ins are, at a minimum, three files that are packaged together in a jar file. You create the following three files (Figure 9-9 on page 491):

```
yourpackage.YourIConnectorPlugin.java
yourpackage.YourConnectorPlugin.java
yourpackage.YourIConnectorPluginDescriptor.java
```

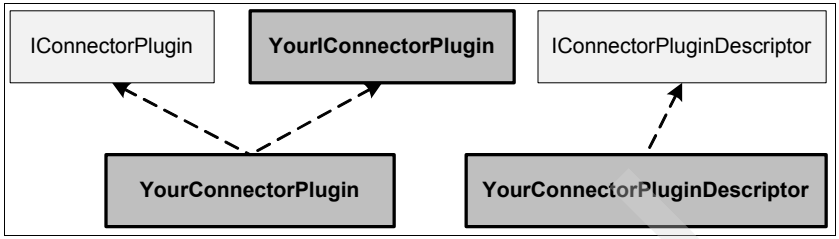



*Figure 9-9 Three classes to create for connector plug-ins*

### *YourIConnectorPlugin.java*

The first class to create defines the methods that provide the functionality that you need to plug in to a connector. It does not need to extend or implement any classes.

It merely gives the method signatures, as shown in Example 9-21. These methods are defined in the next Java class that we discuss.

*Example 9-21 Snippet of code from sample IBasicConnectorPlugin.java*

```
package com.ibm.eci.examples.connector.plugins.basic;

import com.venetica.vbr.client.VeniceBridgeException;

public interface IBasicConnectorPlugin {

   // Property used
   public static final String NAME_PREFIX_PROP_KEY =
      "plugin.name.prefix";

   /**
    * Simple example method.
    * Returns the String that is passed in.
    */
   public String echoMessage(String msg) throws VeniceBridgeException;

   ...

   /**
    * Modified getRepoItem method.  If the plugin.name.prefix config
    * property exists the name of the returned item is modified.
    *
    */
   public RepoItem getRepoItem(RepoItemHandle handle)
      throws ItemNotFoundException, RemoteException,
```

Archived

```
        PermissionDeniedException, VeniceBridgeException;
}
```

### *YourConnectorPlugin.java*

*YourConnectorPlugin* is the second class that must be created and implements both the interface in Example 9-21 on page 491 and the com.venetica.vbr.plugin.IConnectorPlugin interface from the Java SPI. Example 9-22 shows that this class defines the methods specified in IBasicConnectorPlugin, as well as the methods from IConnectorPlugin.

*Example 9-22 Code snippet from sample BasicConnectorPlugin.java*

```
package com.ibm.eci.examples.connector.plugins.basic;

public class BasicConnectorPlugin implements IConnectorPlugin,
      IBasicConnectorPlugin {

// A handle to the connector
   private IBridge _connector = null;
   private Map _sessionObjects = null;

   // Method declared in IConnectorPlugin
   public void destroy() {
      // Free any resources we are using
      _connector = null;
      _sessionObjects = null;
   }

   // Method declared in IConnectorPlugin
   public IConnectorPluginDescriptor getDescriptor() {
      return new BasicConnectorPluginDescriptor();
   }

   // Method declared in IConnectorPlugin
   public void initialize(IBridge connector, Map sessionObjects) {
      // Keep the connector handle for later.
      _connector = connector;

      // Additional information the connector is providing to us
      _sessionObjects = sessionObjects;
   }

   // Method declared in IConnectorPlugin
   public boolean isInitialized() {
      // Do we have everything we need?
```

```
        return (_connector != null);
    }

    /*****************************************************************
     * Plugin methods from IBasicConnectorPlugin
     *****************************************************************/

    public String echoMessage(String msg) throws VeniceBridgeException {
        // This example simply returns the passed in parameter.
        return msg;
    }

    ...

    public RepoItem getRepoItem(RepoItemHandle handle)
        throws ItemNotFoundException, RemoteException,
            PermissionDeniedException, VeniceBridgeException {

        RepoItem item = _connector.getRepoItem(handle);

        // Modify the name if we are told to
        String prefix = getPluginProperty(NAME_PREFIX_PROP_KEY);
        if (prefix != null) {
            item.setName(prefix + item.getName());
        }

        return item;
    }

    ...

}
```

While these methods do not necessarily show actual examples, they effectively illustrate how you can add new methods, which can be made available to an Content Integrator client. With a repository API, the client can send requests to your plug-in, which are passed along to the repository. Use the session object that is passed to the initialize(IBridge, Map) method in the Map object to communicate with your repository.

### *YourConnectorPluginDescriptor.java*

*YourConnectorPluginDescriptor* is the last of the three classes to create. It implements the com.venectica.vbr.plugin.IConnectorPluginDescriptor interface,

which declares several methods that can be used to describe your connector plug-in, as shown in Example 9-23.

*Example 9-23 Sample BasicConnectorPluginDescriptor*

```
package com.ibm.eci.examples.connector.plugins.basic;

import java.util.Properties;
import com.venetica.vbr.configuration.BridgeConfig;
import com.venetica.vbr.plugins.IConnectorPluginDescriptor;

public class BasicConnectorPluginDescriptor implements
java.io.Serializable, IConnectorPluginDescriptor {

   public Class getPluginClass() {
      return BasicConnectorPlugin.class;
   }

   public Properties getDefaultPluginProperties() {
      Properties props = new Properties();
      // No default properties being set
      return props;
   }

   public String getDescription() {
      return "Simple plugin example. Provides a starting point for new”
            + “ connector plugins.";
   }

   public String getName() {
      return "BasicConnectorPlugin";
   }

   public String getOwner() {
      return "IBM";
   }

   public String[] getSupportedConnectors() {
      return new String[] {BridgeConfig.SAMPLE_FILE_SYSTEM_CONNECTOR};
   }

   public Class getPluginInterface() {
      return IBasicConnectorPlugin.class;
   }
```

Archived

```
   public String getPluginVersion() {
      return "1.0";
   }

}
```

Notice that the getPluginInterface() method returns your newly created connector plug-in interface class and not the IConnectorPlugin class. The API uses your interface along with the IConnectorPlugin class to have a list of the methods that are available to the client.

### 9.3.3 Packaging a connector plug-in

The process to build your plug-in is simple. First, you create a manifest file with an entry to identify the class that implements the com.venectica.vbr.plugin.IConnectorPlugin interface. Example 9-24 shows a sample manifest file that is available at this location:

*ICI_HOME*/docs/examples/java/PLUGIN_MANIFEST.MF

*Example 9-24 Sample manifest file*

```
manifest-Version: 1.0
Created-By: IBM Corporation
Copyright-Notice: Licensed Materials - Property of IBM 5724-J31 (c)
Copyright IBM Corp. 2000, 2006 All Rights Reserved. US Government Users
Restricted Rights - Use, duplication or disclosure restricted by GSA
ADP Schedule Contract with IBM Corp.
Class-Path: @required_jars@

Name: com/venetica/vbr
Specification-Title: Content Integrator
Specification-Version: 8.5
Specification-Vendor: IBM Corporation
Implementation-Title: com.venetica.vbr
Implementation-Version: @build_number@

Name: com/ibm/eci/examples/plugins
Plugin-Class: com.ibm.eci.examples.connectors.plugins.basic.BasicConne
ctorPlugin
```

The last two lines of the manifest file in Example 9-24 are the important lines. The Plugin-Class entry expects the fully qualified name of your plug-in class.

Next, you package your class files and the manifest file together in a jar file. There is a sample Apache Ant script to build your connector plug-in jar available at this location:

*ICI_HOME*/docs/examples/java/buildExamplePlugins.xml

Now that you have your connector plug-in jar, you need to deploy it to one or two locations depending on where your connectors are deployed. In every case, you must deploy the plug-in jar file to the following location where Content Integrator is installed:

*ICI_HOME*/plugins

If your connector is deployed outside of the installation of Content Integrator, the plug-in must be deployed to the plugins directory there, as well. Correctly deploying your connector plug-in insures that all Java virtual machines (JVMs) have access to the new classes.

### 9.3.4 Configuring a connector plug-in

The client application logs in to the appropriate connector. Next, it obtains a proxy connection to the desired connector plug-in by calling the Repository.getPluginProxy(String) method. This method uses the connector plug-in's name as a parameter, which is defined in the getName() method of your descriptor class. An object is returned that must be cast to the user-defined plug-in interface. If the connector plug-in cannot be found, an error is returned. If you are not sure of the name of a connector plug-in, you can call Repository.getPluginDescriptors() to get an array of IConnectorPluginDescriptor files that are available to the client. For an example of calling a connector plug-in from a client, look at this sample:

*ICI_HOME*/docs/examples/java/commandline/ExecConnectorPlugin.java

## 9.4 Summary

This chapter shows you how to build a custom connector (new or extended) or a connector plug-in, depending on the needs of your environment. There are several chapters in this book that can be useful to a connector developer or anyone who wants an in-depth understanding of Content Integrator capability and design:

- Chapter 6, "Content Integrator architecture" on page 291 covers the architecture of Content Integrator and gives you a better understanding of where a connector fits into the larger picture.

- Chapter 8, “Understanding the client APIs” on page 385 explains how Content Integrator client applications are written.
- Chapter 10, “Content Integrator industry solutions” on page 499 covers many of the actual industry solutions that have been implemented and that include Content Integrator.

The following Web sites provide additional information that is related to Content Integrator and Enterprise Content Management:

- Content Integrator Information Center contains overviews and details of all of the features and abilities of Content Integrator:

  http://publib.boulder.ibm.com/infocenter/ce/v8r5/index.jsp

- IBM Lab Services can best assist clients in their efforts to create a needed custom connector or a connector plug-in for their environment:

  http://www.ibm.com/software/data/services/cm.html

- Content Integrator API and SPI contain JavaDocs for all of the available classes for client and connector development:

  *ICI_HOME*/docs/integrate/api/index.html
  *ICI_HOME*/docs/integrate/spi/index.hml

Archived

Archived

10

# Content Integrator industry solutions

In this chapter, we present a number of scenarios where developers use IBM Content Integrator in industry to solve a range of business needs. Beginning with examples of the direct use of IBM Content Integrator, we present cases from the industry where IBM Content Integrator helps to integrate business application needs with their content management systems. In addition, we illustrate examples of embedding IBM Content Integrator with other product implementations, extending the scope and value for their target users.

We cover the following topics in this chapter:

- Direct use of IBM Content Integrator:
  - Case one: Content access and federated search platform
  - Case two: Document exchange platform
  - Case three: Federated records solution
- Embedded use of IBM Content Integrator:
  - Case one: IBM Omnifind Enterprise Search
  - Case two: Licensing and Case management solution

© Copyright IBM Corp. 2010. All rights reserved.

## 10.1 Direct use of IBM Content Integrator

A significant percentage of enterprise business content exists in an unstructured format across various data sources or content management systems. Represented as billing statements, business contracts, marketing collateral, legal documents, and Web content, individuals use this data in customer service, multichannel marketing, and corporate communication. This data becomes increasingly more valuable each and every day. In fact, businesses cite their ability to locate and use this unstructured information as being a critical component of the success of the business.

Whether through organic growth, or through the natural processes of mergers and acquisitions, the challenges, which are presented with valuable business data that is segregated across multiple content management systems in silos, often plague organizations desiring to make use of this information. In time, it can become extremely difficult for a particular line of business to have an holistic view of the data that is necessary to make informed decisions. Figure 10-1 presents the ease at which this complexity can grow. For example, a customer service portal needs statement data from the report management system, loan application information from a document management system, and scanned documents from a network file system. Without the benefit of content integration technology, point-to-point integration is required by the customer service portal.

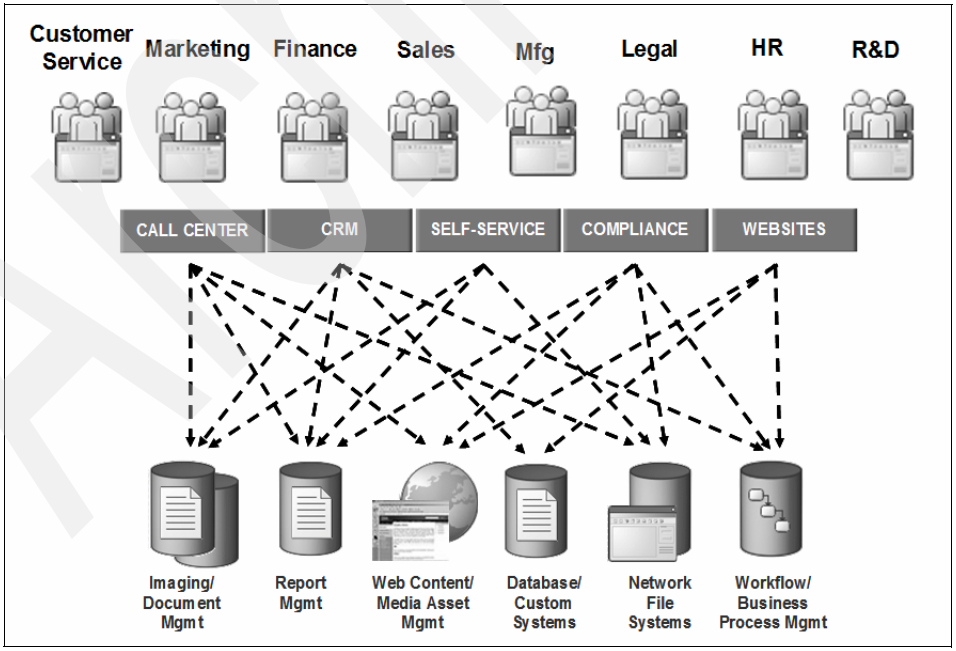



*Figure 10-1 A mature enterprise content environment*

Without an organized and systematic approach to face the challenges of disparate content management systems, businesses must face the following challenges:

- Integration efforts are costly, lengthy, and often unrepeatable.
- Developer expertise is required for each content management system application programming interface (API).
- Projects experience higher ongoing maintenance costs that are related to upgrades and consolidations at the data layer.
- Organizations must divert their development resources away from solving business problems while application integration efforts are underway.

The design philosophies of IBM Content Integrator alleviate the complexities that are presented by disparate content management systems to growing businesses. IBM Content Integrator's easily consumable API provides a suitable middleware approach to address many point-to-point integration concerns. Acting as the broker between applications and the data that they consume, IBM Content Integrator can provide seamless, real-time, and bidirectional access directly across various content management systems. IBM Content Integrator requires no data migration; previous investments are secure.

Three case studies follow that illustrate the direct use of IBM Content Integrator.

### 10.1.1 Case one: Content access and federated search platform

The first case that we present is a common scenario that involves a large banking institution that constructed its own content access and federated search platform using IBM Content Integrator.

#### Company background

The bank that is presented in this case offers corporate and retail banking, asset and wealth management, and capital market and securities brokerage services and products.

The bank has grown through several mergers and acquisitions in the recent past. Of course, each acquired company maintained its own preferred set of content management systems to store millions of scanned documents, checks, and legal forms in these various systems, often using their native tools. This disparate storage of the bank's critical information hindered customer service and the bank's ability to capture additional market share.

Archived

## Business needs

The search for a content integration solution began with the bank's first large merger. The bank recognized early the challenges that were presented with vital data dispersed across multiple content repositories. Without manual steps, it was difficult for the various lines of business to view customer information that was stored in disparate systems. Inaccessible data was not an acceptable mode of operation. It became critical to find a solution.

The bank began exploring solution options. Key requirements surfaced:

- Eliminate the need to rewrite applications that access separate content repositories.
- Insure the ability to search for information across all systems.
- Insure the ability to view or convert document types into preferred formats.
- Reduce the time that is necessary to integrate disparate systems in the future.

## Solution overview

After ruling out data consolidation altogether, and after exploring other information integration tools, the bank decided to use IBM Content Integrator to implement its content access and federated search platform. This integration platform unites the bank's content management systems, providing a single point of access for its customer service, brokerage, and workflow applications. The services that were provided include content search, content retrieval, and the storage and maintenance of content, as well as administration services, such as logging, alerts, and reporting.

Today, the bank's content integration platform integrates five content management systems, supports eight lines of business, and more than 20 applications, including opening new accounts, annuities services, commercial lending, loan processing, and retail credit servicing. The bank's platform now supports over 15 million transactions per month and more than 500,000 distinct users per month, with a 24x7 high availability environment.

As presented in Figure 10-2 on page 503, the content integration platform provides a unified programming interface among content management systems and a host of business applications. Offered as a managed service to internal lines of business, the content integration platform receives requests from line of business applications that are targeting the data source connections maintained by IBM Content Integrator. These requests are then properly distributed to the correct content management system through both product-delivered connectors, as well as a couple of customer-built connectors using the IBM Content Integrator connector service provider interface (SPI). Load balancing across multiple computing environments is performed during peak usage, as necessary.

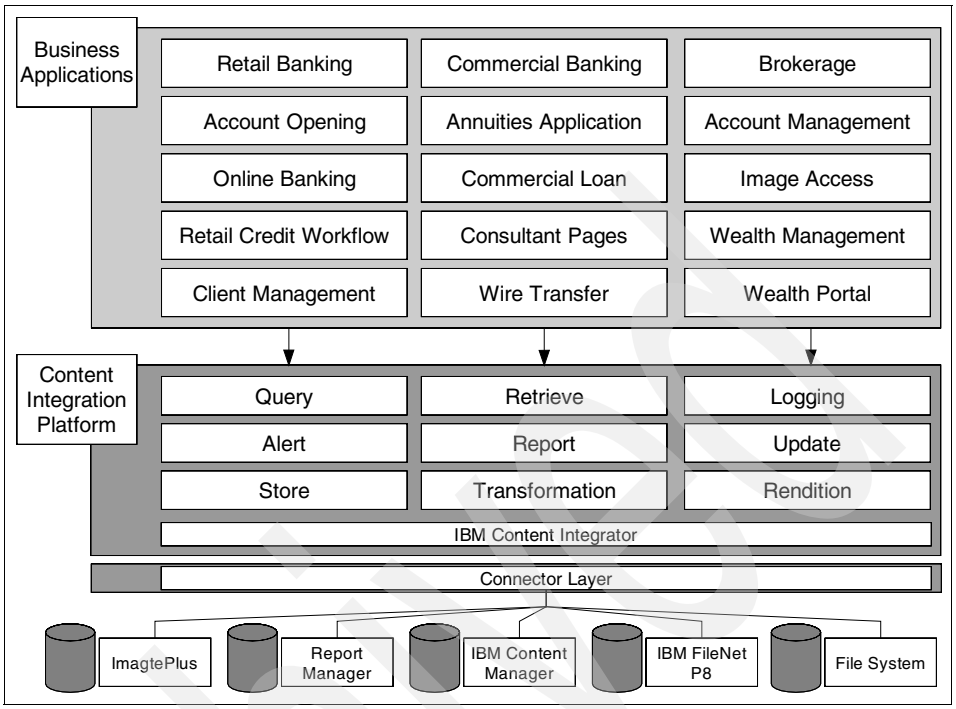



*Figure 10-2 Logical model of the content integration platform architecture*

Using this content integration platform environment provides several advantages:

- Business units can scan documents directly into various content management systems.
- Customer statements are stored as Advanced Function Print (AFP) documents in the report management system.
- Connectors integrate various image repositories with the content integration platform.
- Vendor APIs are abstracted into an XML interface that leverages Web Services (SOAP/HTTP).
- The platform provides a view converter framework to handle document conversion into PDF format.
- All line of business application developers implement to the content integration platform interface.
- From their desktops, users can access documents and account statements from within their browser-based applications.

Archived

- Developers can incorporate new connectors into the content integration platform at any point in time.

## Supported applications

Presently, the bank's content integration platform supports an array of applications across eight lines of business. Next, we discuss several examples of the types of implementations that take advantage of the content integration platform and IBM Content Integrator.

### *Retail loan application*

New accounts requesting individual loans are managed by the retail loan application. Customers sign the hardcopy versions of loan applications, which are then scanned into either an IBM ImagePlus® or an IBM FileNet Content Manager system. These loan applications are then routed to the proper individuals for verification and approval.

An acquisition by the bank brought along a separate content management system that was full of customer loan applications. Because the time to migrate the millions of documents from IBM ImagePlus into IBM FileNet P8 was prohibitive, and because there was an immediate need to be able to access the information that was stored in IBM ImagePlus, the bank selected to use IBM Content Integrator. Following a hybrid approach, only the necessary and sufficient indexes were migrated from IBM ImagePlus to FileNet P8. Additionally, the bank wanted the FileNet system to manage document access security.

Using IBM Content Integrator's API, the original FileNet-based loan application did not need any changes. A connector to IBM ImagePlus was built by the bank using IBM Content Integrator's connector SPI. The retail loan application issues searches against the IBM FileNet Content Manager system only. Based on the results of these search requests, documents are retrieved from the proper system. This approach largely reduced the expected migration effort while making all documents from either content management system available to the retail loan application.

### *Commercial loan application*

Similar to the retail loan application, the commercial loan application handles new account openings for commercial loans. Wanting to move away from the older IBM ImagePlus system to a newer IBM Content Manager installation, the bank uses IBM Content Integrator to communicate with both systems. In response to user search requests, background processes migrate the necessary requested information to IBM Content Manager without any disruption to the user or to the business.

### *Bank securities broker front office and workflow application*

The bank securities workflow application is backed by many content management systems. Because the application requires that all document formats are in PDF form, the content integration platform and IBM Content Integrator convert the native content formats into the desired format. The content integration platform and IBM Content Integrator convert the MO:DCA, IOCA, PTOCA, TIF, AFP, and TXT document formats to PDF upon retrieval so that the documents are presented properly in the securities workflow application.

### *Online bank statement access*

The bank provides an online statement service for paperless statements to more than 500,000 customers. Both bank statements and check images are available for viewing through the service, each type of document coming from a separate content management system. The bank uses IBM Content Integrator to deliver both pieces of information to the customer in a single request.

## Business benefits

The content integration platform and IBM Content Integrator have saved the bank $2.3 million within the first two-year period after the implementation. The number of requests for content has increased 50 fold. Continuing to grow, the bank plans to add more IBM Content Integrator connectors to support additional users that are targeted for inclusion into the platform.

Since implementation, the content integration platform has had these successes:

- Rapid development of new applications to meet business needs
- Flexible infrastructure that is capable of meeting scalability needs
- Bank agents provided seamless access to customer documents

## Lessons learned

Based on the bank's deployment experiences after a series of mergers and acquisitions, a systematic approach to handling and transitioning disparate content management systems helped satisfy business needs. Consider these areas:

- Understand and evaluate all available options for cost and effort:
  - Complete data conversion and migration from one system to another
  - Partial data conversion and migration
  - Point-to-point data conversion and migration
  - No conversion or migration by using IBM Content Integrator
- Understand business needs, time line, and performance requirements.
- Fully utilize product features that are provided by IBM Content Integrator.
- Use a progressive approach: Start small to gain user acceptance.

Archived

## 10.1.2 Case two: Document exchange platform

A large, global petroleum company is the focus of our second case. This petroleum company built their own document exchange platform using IBM Content Integrator.

### Company background

The petroleum company is a leading manufacturer, distributor, and marketer of refined petroleum products.

### Business needs

Like many manufacturing companies, the petroleum company produces thousands of engineering drawings and related documentation. The majority of these documents need to be exchanged with business partners on a daily basis. In an effort to shed a manual and error-prone distribution process, the petroleum company decided to build an automated and more efficient document exchange platform.

The company wanted the new system to have these capabilities:

- The ability to exchange documents between separate content management systems. The petroleum company uses Open Text Livelink and the partners use EMC Documentum, Hummingbird Document Management, and various IBM systems.
- The ability to exchange documents in response to metadata changes
- The ability to transfer documents in bulk
- The ability to automatically receive documents upon completion from other users
- The ability to notify other users of a document's transfer status (exchanged, in process, rejected, or successfully transferred)
- The ability to transfer documents between disparate private networks

### Solution overview

The petroleum company built a document exchange platform using IBM Content Integrator to transfer engineering drawings and related documentation among disparate participating content management systems.

The document exchange platform monitors document metadata in source content management systems. When there is a change detected in the metadata that meets specified selection criteria, the content is selected for transfer to a target content management system. The selected content, along with its metadata, is then transferred using a secured virtual private network (VPN),

Archived

ingested either as an updated version of an existing document or as a new version of a new document.

The following features of IBM Content Integrator were used in developing the document exchange platform:

- Access services
- Content management system connectors
- Subscription Event Services rules broker Enterprise JavaBeans (EJB)
- Data maps
- RMI/SOAP Proxy Connector
- Log4j application logging (FileAppender and JDBCAppender)

Using these features and components of IBM Content Integrator, the document exchange platform can now perform these functions:

- Handle queuing multiple documents for transmission.
- Create document exchange requests based on configurable XML rules.
- Move documents between source and target content management systems.
- Cross-reference metadata properties between systems using data maps.
- Communicate with many systems using IBM Content Integrator connectors.
- Use VPN connections on remote networks by using RMI, SOAP, and HTTP.

#### Business benefits

The petroleum company has saved money and received an increase in productivity since implementing and deploying the content exchange platform that was built using IBM Content Integrator. Additionally, using an automated transfer system has largely reduced the errors that were introduced with a manual process.

### 10.1.3 Case three: Federated records solution

The third case that we present is a scenario that happens with more frequency. A leading financial management and advisory company has implemented a federated records solution with the help of IBM products, including IBM Content Integrator.

#### Company background

The company that is presented in this case is a financial management and advisory company that provides investment banking services and financial advice to its customers.

Archived

## Business needs

The financial company wanted to implement an enterprise-wide electronic records management system that automatically applies retention schedules to documents in various content management systems. The solution needed to provide full records life cycle management control. The financial company has these specific goals and capabilities for the final solution:

- Obtain US DoD 5015.02 V2 certification
- Provide a centralized system to declare, retain, and dispose of records based on the corporate records retention policy
- Provide tracking and audit capability of corporate documents and e-mails
- Be able to declare records manually or without user interaction
- Integrate records declaration within a workflow process
- Apply legal holds on records
- Facilitate the efficient discovery of corporate documents for litigation support
- Be able to integrate with a paper-based records solution
- Enable e-mail as records to be managed
- Use federation capability to manage records across disparate content management systems
- Use federated search capability to search for documents before record declaration
- Be able to enable records across the following content management systems:
  - IBM FileNet Content Manager
  - EMC Documentum
  - IBM Content Manager OnDemand
  - Microsoft Windows network file systems

After evaluating a number of solutions, the financial company decided to adopt IBM Enterprise Records (formerly known as IBM FileNet Records Manager) as the central records management engine for the entire company. The financial company used IBM Content Integrator to provide integration, allowing records control for content that is stored in EMC Documentum and IBM Content Manager OnDemand.

## Solution overview

The financial company selected the following components to make up the solution after identifying all of the business requirements:

- IBM Enterprise Records (formerly known as IBM FileNet Records Manager)
- IBM FileNet Content Manager

- IBM FileNet P8 Content Federation Services (CFS)
- IBM Content Integrator
- IBM Content Collector for File Systems

The financial company uses IBM CFS and IBM Content Integrator together to populate the FileNet P8 master catalog with the necessary metadata that is required to manage federated content in the EMC Documentum content management system. When necessary, the Enterprise Records triggers record life cycle actions (lockdown, disposition, and holds), and IBM Content Integrator initiates these actions against EMC Documentum.

Using this approach, no data migration is necessary from EMC Documentum to IBM FileNet P8. IBM CFS and IBM Content Integrator work together to manage records in the native content management system.

#### Business benefits

The financial company gained these benefits from this approach:

- Providing consistent records policies across distributed content management systems
- Eliminating costly records discovery using federated search capabilities
- Addressing dynamic organization needs as mergers and acquisitions occur
- Allowing content management systems to be added, removed, and modified without disrupting the management or records or risking non-compliance

## 10.2 Embedded use of IBM Content Integrator

Companies can often expand solutions, which are developed to solve common business needs, for greater coverage or to provide a larger user base, simply by including access to content management systems that were originally unavailable to the solution. We have covered one example of the embedded use of IBM Content Integrator in detail in Chapter 5, “Content Federation Services implementation using Content Integrator” on page 189. Because embedded use is growing more common in the industry, we present two additional embedded uses of IBM Content Integrator.

### 10.2.1 Case one: IBM Omnifind Enterprise Search

The first IBM Content Integrator embedded use case that we present involves IBM Omnifind Enterprise search solution.

### Solution overview

IBM Omnifind Enterprise search is a middleware layer between user applications and various enterprise content management systems. It provides a search-based application that is able to access all enterprise information using a single interface. Using keyword or phrase searches, users of IBM Omnifind Enterprise search are quickly presented with the most relevant information across the user's enterprise.

IBM Omnifind Enterprise search consists of four major technologies: crawling, parsing, indexing, and searching. Crawlers extract content from their native data sources, and the index server parses and analyzes the content to build an index, while the search component processes search requests to return relevant results using the indexes that are built.

IBM Content Integrator, and a subset of the product connectors, is embedded as a portion of the crawler technology. We can use these IBM Content Integrator connectors:

- EMC Documentum
- Hummingbird Document Management (DM)
- IBM Content Manager
- IBM Content Manager OnDemand (CMOD)
- IBM FileNet Content Manager
- IBM WebSphere Portal Document Manager
- Open Text Livelink
- Microsoft Windows SharePoint Services

IBM Content Integrator provides this functionality for IBM Omnifind Enterprise Search:

- It extends crawl options with product connectors.
- You can build custom connectors easily and add them to the deployment.
- It supports real-time and bidirectional access.
- It uses native content management functionality, such as full-text search.
- It converts native content into browser-readable formats dynamically.
- You can search across multiple content management systems.
- You have single sign-on (SSO) access to multiple content management systems.

## 10.2.2 Case two: Licensing and Case management solution

Our second IBM Content Integrator embedded use case is a Licensing and Case management solution.

Archived

## Solution overview

A Licensing and Case management solution is a document management solution for local, municipal, state, and federal governments that provides specific modules for managing licenses and the building permit process. The solution automates the processing of licenses, permits, and registrations for applications through official inspections. This solution tracks all activities that are associated with the process, including licensing inspections, complaints or regulatory investigations, hearings and legal actions, board decisions, and others.

The Licensing and Case management solution is built in three components:

- Presentation layer: HTML V4.0 browser interface with Microsoft Internet Explorer V7 leveraging AJAX to optimize the user experience
- Application layer: Java 2 Platform, Enterprise Edition (J2EE) and service-oriented architecture (SOA)
- Database layer: Oracle and Microsoft SQL Server

When a customer submits a building permit for approval, the permit is kept in the agency's own content management system and begins a document approval process. This process requires the document management capabilities:

- Check in
- Check out
- Workflow
- Annotation
- Search
- Retrieval
- Document conversion

The solution's own database store is unable to manage the workflow if the agency has a separate content management system to store those permits.

To overcome the complexities arising from disparate content management systems, we embedded IBM Content Integrator into the solution's application layer to send requests, such as search, retrieve, and create, to the IBM Content Integrator Java API using a Web Services Description Language (WSDL). This use of IBM Content Integrator complements the solution with repository connectors, providing the ability to manage the permit process regardless of the agency's content management system type.

Archived

Archived

A

# Planning and sizing the Content Integrator system

This appendix presents the highlight of the white paper “IBM Content Integrator Sizing Guidance” made available after the release of IBM Content Integrator Version 8.5. The goal of the paper is to provide high-level guidance for Content Integrator system sizing and environment configuration.

This paper is organized in the following two sections:

- Description of variances applicable to Content Integrator
- Strategy for system planning

© Copyright IBM Corp. 2010. All rights reserved.

# Overview and purpose

This document provides high-level guidance for Content Integrator system sizing and environment configuration. Content Integrator is Enterprise Content Integration middleware and, as such, is deployed in scenarios where application needs are dependent on customer usage and deployment. Customer specific application needs and the myriad of deployment options make generalized system sizing and configurations variable.

A description of common variance points is presented, followed by a description of an overall strategy useful in creating an estimate of the hardware required.

*Table A-1 Terminology*

| Term | Definition |
|---|---|
| Repository | An enterprise content management system, such as IBM FileNet Content Manager |
| Connector | *Connectors* act as adapters, translating the Content Integrator integration API calls to the vendor-specific APIs of back-end repositories. |
| RMI proxy connector | There are several reasons to consider a configuration that includes running the connector outside of the application server. In these configurations, the RMI proxy connector serves as the host process for the connector and provides remote access to its features using Java's Remote Method Invocation (RMI) protocol. |
| Content Integrator | Content Integrator is the leading infrastructure for enterprise content integration. With Content Integrator, global organizations enable portals, collaborative applications, customer relationship management and other key applications to work with distributed content throughout the extended enterprise. |
| Session pool | A *session pool* is a collection of Content Integrator User objects that have already been logged in. This is useful in situations where common user credentials are used by a group of users to perform operations. The overhead associated with checking out a User from the pool is generally lower than the overhead of performing a login. |

Archived

# Variances applicable to Content Integrator

There are choices a system architect can make that affect Content Integrator performance. Hardware selection, operating system virtualization, and even network speeds can significantly impact overall system performance. Additionally, the types of operations that an end application uses may affect perceived performance. The deployment of Content Integrator should be chosen to balance application needs and hardware requirements.

## Hardware

Hardware choices affect the overall system performance in the most noticeable fashion. Both the system processor capabilities and the amount of memory available must be taken into consideration, along with additional processing requirements of the system. For example, if the system hosting Content Integrator also serves other purposes, additional memory and processor power may be required.

### CPU

CPU evolution now focuses on multithreading and multiple cores to achieve performance increases. Content Integrator relies on integer operations primarily, so when choosing a processor, more weight should be given to integer performance than floating point performance. In some deployment scenarios, Content Integrator can take great advantage of multiple CPU cores and native multithreading support.

Content Integrator can take the greatest advantage of parallelism with the RMI proxy connector deployment scenarios. In RMI connector deployment scenarios, a separate operating system thread is spawned to handle each user session. Under a heavy load, consider increasing CPU threads (via additional processors or multi-threaded cores) to achieve higher throughput.

### Memory

Memory is also an important consideration with Content Integrator, especially in RMI connector deployment scenarios. The connectors employed by Content Integrator maintain user security and state by isolating each user's session. This necessarily increases the amount of memory required for parallel loads. Each user session will increase the amount of memory required by Content Integrator, and the amount of memory required will vary based on the connector in use. It is important to insure that enough memory is available in the Java heap space to handle the expected concurrent load.

Archived

When handling large loads, or when load balancing is desired, consider using multiple RMI connectors. For 32-bit operating systems and JVMs, especially on the Windows platform, there is a 2 GB limit on the amount of memory addressable by a single process. It is advisable to use multiple RMI connectors in a load balanced configuration with heap sizes optimized for your work load, garbage collection performance requirements, and available system memory.

## Virtualization

Although great strides are being made in virtualization technology on an almost daily basis, the greatest performance is still achieved using “bare metal”. Approach virtualization with care, especially in production environments. Be sure to provide additional processing power if you choose to use virtualization.

## Deployment

Equally important to hardware considerations, the deployment architecture of Content Integrator is critical to the successful implementation of a project. Content Integrator can exist as a native library fully embedded within an application, provided that the connector involved utilizes a Java API for communication with the repository. However, for J2EE applications making use of Content Integrator, some connectors should only be used in RMI connector mode. This is due to the fact that some connectors utilize the Java Native Interface to interact with a repository, which violates the J2EE specifications - if native components crash or leak memory, the J2EE container would be affected.

### Single JVM deployment/complete embed

The primary benefit of a fully embedded Content Integrator approach is speed. As in all deployment configurations, a single-JVM application/Content Integrator deployment makes use of Content Integrator as though it were a local library. Calls to Content Integrator are translated to native repository operations and issued with the fewest number of network hops possible. See Figure A-1 on page 517.

Archived

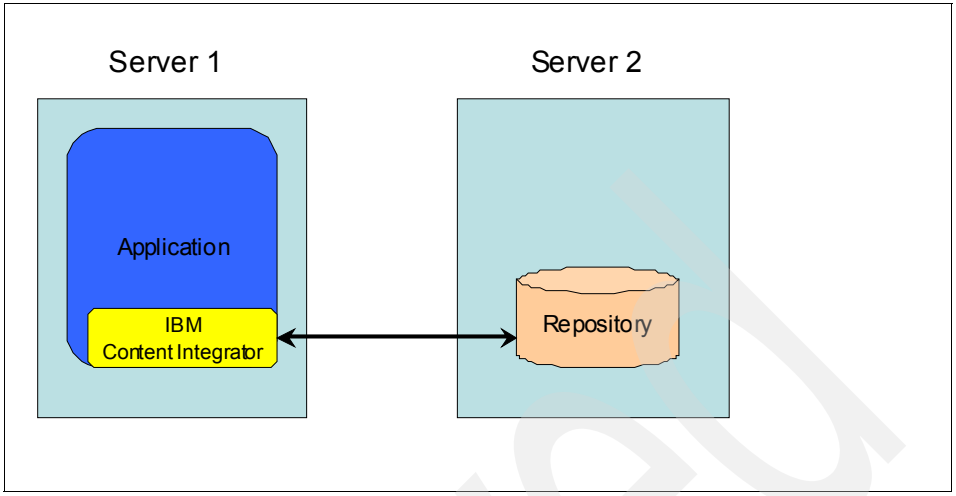



*Figure A-1 Compete embedded deployment configuration*

The drawback to this deployment configuration is that native repository libraries must also be packaged within the application for Content Integrator to use. This adds to the overall size of the application packaging. If the application is delivered to a Web browser, this can slow the delivery directly proportional to the total size of the packaging. Additionally, there is no failover or load balancing provided by Content Integrator in this scenario. The containing application would need to provide it if it was a system requirement.

## RMI proxy connector deployment

The RMI proxy connector deployment configuration allows for centralized location and configuration of the communication between Content Integrator and the repository. Client applications still communicate to Content Integrator as though it were a local library, but Content Integrator utilizes Remote Method Invocation (RMI) to call methods on its connectors. The connectors are hosted by a secondary JVM. This allows configuration of repository-specific libraries, paths, and native components outside of the application's JVM, which reduces the application's exposure to crashes caused by native communication components. See Figure A-2 on page 518.

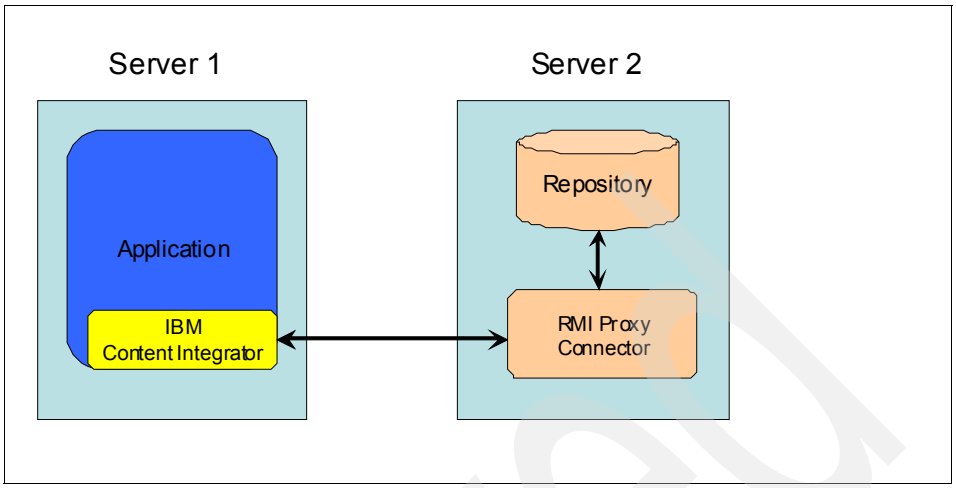



*Figure A-2 RMI proxy connector deployment configuration*

This deployment configuration requires a JVM to be running outside of the client application, and incurs network delays between the Content Integrator client API and the connector. However, this configuration also allows the repository-specific code to be as close as possible to the repository. In the case of a “chatty” repository communication layer, this co-location of connector/repository can actually improve speed in relation to a complete embed configuration.

This configuration also provides load balancing and system failover capabilities. Multiple RMI proxy connectors can be configured to communicate with the same repository. Content Integrator will then “round robin” across the configured RMI proxy connectors to achieve basic load balancing. In the event that an RMI proxy connector crashes, sessions that are active on that connector are lost. The overall system will compensate for the lost connector by placing subsequent sessions on the remaining active RMI proxy connectors.

## Profile of Content Integrator operations used

In an ideal world, all operations take the same amount of time to complete, and sizing does not have to take into account the mix of operations used. Unfortunately, this situation is not the case, and even the same operations take differing amounts of time on separate repositories. Some repositories are optimized for query, others for retrieval. Any sizing effort will need to have a thorough knowledge of the types of operations involved by the application and users, and the relative speeds of the operations in the environment.

It is recommended that a detailed analysis is made of the typical use cases required, and a “profile” generated. This profile will indicate how often each type of operation is to be used, and in what order.

### Java version

Java is a dynamic language, and performance optimizations are introduced from time to time. Generally speaking, newer versions of the JVM should perform better than older ones, and where performance degrades, it is generally to correct behavioral issues.

### Repository speed

Repositories are not all created equally. Some repositories are optimized for search capabilities while others are optimized for retrieval. An example of this would be a connector written against a search engine. A search engine vendor necessarily focuses on search, and may not support scalable architectures with regards to retrieval.

## Strategy for system planning

When planning a system for Content Integrator, the primary factors to consider are desired headroom and the performance of Content Integrator with respect to the workload required. This section will set out concrete steps to perform to get accurate performance characteristics. In general, size the system such that Content Integrator operations are constantly being executed without waiting on CPU time. Your IBM representative can help you with system sizing.

### Determine the load

The load on the system consists of the size of the user population, the activities they perform, and the frequency with which they perform them. Load can greatly increase due to “impatient user syndrome,” in which a user repeatedly requests operations. Define acceptable wait time based on your user population. Be sure to consider any batch-type operations that may be required of the system. Batch operations may need to be throttled to insure that a backlog does not occur.

Archived

## Determine deployment configuration

Decide which deployment configuration suits the application best. If centralized configuration of repository communication is desired, or if the application is running in a J2EE application server and the repository has native components, choose the RMI proxy connector configuration.

Most of the translation work is performed at the Content Integrator connector layer. To achieve maximum throughput, insure that each Content Integrator operation will have an available execution core on the application host system. Time spent waiting on the CPU to service the Content Integrator thread lowers the overall throughput.

Content Integrator has a way to speed up login operations in the event that multiple users share the same credentials. The Content Integrator Server Mode session pool allows these users to be checked out of a pool of logged in Content Integrator User objects rather than having to go through the normal login process. This pooling mechanism can be used with any deployment configuration, but it does require a pool server process to be running from which to check out the Content Integrator Users. Calls made on pool Users take the same amount of time to perform all other operations as a User object logged in by normal means.

Content Integrator Local Session Pools may be used in a similar fashion when there are no common credentials if the application requires frequent logins.

## Determine application usage profile

Inspect the Content Integrator operations required by the users of the application. A common profile would be to perform a query, followed by a retrieval of metadata, and potentially retrieval of a document. The best profile data will be obtained by running simulations of the use case on existing hardware, and scaling the sizing requirements according to the results of the tests.

Once the mix and frequency of Content Integrator operations are known, it should be possible to construct a model of processor utilization over time. Special attention should be given to peak processing demands, as these can overwhelm the system if not taken into consideration.

## Determine desired headroom

Once the deployment configuration and application usage profile are known, it should be possible to determine a minimum requirement for processor speed and memory. These sizings should be increased to provide additional headroom

for unexpected surges in demand and future demand expansion. Also, if additional applications will be running on the system, factor their usage patterns into the overall system sizing.

## Determine virtualization needs

If the system is to be run in a virtualized environment, increase the sizing of the system to cover additional processing and memory utilization by the virtualization software. Also include processor utilization by other virtualized systems running on the same host.

Archived

Archived

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

## IBM Redbooks publications

For information about ordering these publications, see “How to get IBM Redbooks publications” on page 524. Note that several of the documents referenced here might be available in softcopy only:

- *IBM FileNet Content Manager Implementation Best Practices and Recommendations,* SG24-7547
- *Introducing IBM FileNet Business Process Manager,* SG24-7509
- *IBM FileNet P8 Platform and Architecture,* SG24-7667
- *Understanding IBM FileNet Records Manager,* SG24-7623

## Other publications

This publication is also relevant as a further information source:

- IBM FileNet Image Services System Tools Reference Manual, CG31-5612
- IBM FileNet Image Services Installation and Configuration Procedures, SC19-2680
- IBM FileNet Content Federation Services for Content Manager OnDemand Configuration Guide, SC19-2711

## Online resources

These Web sites are also relevant as further information sources:

- IBM Content Integrator Information Center

  http://publib.boulder.ibm.com/infocenter/ce/v8r5/index.jsp

Archived

© Copyright IBM Corp. 2010. All rights reserved. 

- IBM Content Integrator Version 8.5.1 publication library

  http://www-01.ibm.com/support/docview.wss?rs=2075&uid=swg27015913#docs

- IBM Content Integrator API and SPI (documentation with the software)

  *ICI_HOME*/docs/integrate/api/index.html
  *ICI_HOME*/docs/integrate/spi/index.hml

- IBM FileNet P8 Platform information page

  http://www.ibm.com/software/data/content-management/filenet-p8-platform

- IBM FileNet P8 Platform product documentation

  http://www.ibm.com/support/docview.wss?rs=3247&uid=swg27010422

  This Web site includes links to all expansion IBM FileNet P8 products.

- IBM FileNet Image Services product documentation

  http://www-01.ibm.com/support/docview.wss?rs=3283&uid=swg27010558

- *P8 Performance Tuning Guide*

  http://www-01.ibm.com/support/docview.wss?rs=3247&uid=swg27010422

- IBM FileNet Content Manager

  http://www.ibm.com/software/data/content-management/filenet-content-manager

- FileNet Content Federation Services documentation

  http://www-01.ibm.com/support/docview.wss?rs=3318&uid=swg27010328

- IBM Enterprise Records (formerly known as IBM FileNet Records Manager)

  http://www-01.ibm.com/software/data/content-management/filenet-records-manager/

- *Compliance and Retention Management with IBM Content Manager OnDemand* white paper

  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101378

## How to get IBM Redbooks publications

You can search for, view, or download IBM Redbooks publications, Redpapers, Webdocs, draft publications and Additional materials, as well as order hardcopy IBM Redbooks publications, at this Web site:

Archived

ibm.com/redbooks

## Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Archived

Archived

# Index


## A
access document 62
access method 4
access services 295, 305
ACL name 340
    combined string 340
add
    IS document 100
Admin Tool
    Content Integrator 215
Administration Tool 305–306, 327–328, 337
Advanced Function Print (AFP) 503
Advanced security model 433
agent
    CE import agent 131
    CFS-IS import agent 54
aggregate the results 34
annot_log table 55
annotation 56, 62
    configure export 104
annotation security mapping
    configure 105
APIs support
    destruction 26
    movement 26
Application Group
    CMOD 146
application programming interface (API) 501, 504, 511
APPLNAME 349
approach
    federation 14
architectural layer 308
architecture
    CFS (using Content Integrator) 202
    Content Federation Services 34, 36, 38
    IBM Content Integrator 292
ARSCFSODWORK table 149, 153
attributes
    metadata 44
audit 25
authentication 300–301, 461
authentication and security 300
authorization 301, 461

## B
barrier 10
BaseBridge 460
BaseConnectorPlugin 490
Basic 421
basic login sequence 392
best practices
    CFS (using Content Integrator) 201
    CFS-CMOD 152
    CFS-IS configuration 136
    records management, CFS (using Content Integrator) 287
bidirectional delete
    CFS-CMOD 151
bidirectional document deletion
    CFS-IS operation 62
BridgeFactory 478
bulk importing
    CFS-IS 130
bundle 461
business process 4
    federated business process management 28–29
    sequence 9
business process management 3, 9, 12
    components 9
    features 13
    federated 28–29

## C
capacity planning
    CFS-IS 136
capture device 5
catalog entries 34
    federated document 45
CE import agent 131
CE log files 121
CFS (using Content Integrator)
    architecture 202
    best practices 201
    common mistakes 271


Archived

© Copyright IBM Corp. 2010. All rights reserved. 

configure security for CM8 204
create 195
database related problems 271
delete 197
deletion 276
Exporter 193
Exporter log 263
Exporter performance 281
exporting 194
FedAdmin log 263
federated document operations 194
federated records management 275
federation rules 247
highly available environment 284
importing 194
installation and configuration 202
JDBC Drivers 272
lockdown 197, 275
logs 262
planning 197
planning consideration 199
retrieve 195
troubleshooting 262
update 195
view federated document 261
view federation rule status 267
when to use 198
CFS Exporter 193
CFS import agent 148
CFS operation
lockdown 41
read 40
update 40
CFS operations
create 40
delete 41, 49
CFS_IMP_QUEUE table 148, 193
CFS-CMOD
best practices 152
bidirectional delete 151
common mistakes 184
create 149
delete 151, 183
exporter 148
exporting 149
federate and lockdown during load 182
federate document 168
federated records management 179
federation database 148
hold 182
import performance 154
Importer 148
importing 149
installation and configuration 154
load document 146
lockdown 150, 181
P8 document class mapping 146
planning 151
records declaration 179
retrieve 150
scheduling utility 148
schema mapper 148
troubleshooting 184
update 150
view federated document 176
CFS-CMOD components 146
CFS-CMOD federation process 148
CFS-IS
best practices 136
bulk importing 130
capacity planning 136
common error, date mismatch 127
common error, field size mismatch 127
common error, menu item mismatch 127
common errors, federation 126
database indexes 138
document class consolidation 137
Exporter 54
exporting 55
federated records management 112
federation database 55
implementing, planning 58
importing 56
improve import performance 130
lockdown 58, 112
metadata change 61
planning 63
retrieval keys 138
schema mapping editor 53
search IS document 102
testing 100
testing, find IS document 101
troubleshooting 121
tuning 130
CFS-IS components 52
CFS-IS import agent 54
CFS-IS operation
bidirectional document deletion 62

Archived

create 56
retrieve 57
update 57
CFSOD export utility 148
configure and run 173
CFSOD exporter utility 154
CFS-operation
delete 58
check in
federated document 45
client application 389
Client Application Programming Interface (API) 386
client locales 406
client side sorting 399
CM8 Item 190
CM8 Item Type 190
CM8 security principal 200
CMOD
Application Group 146
field 146
CMOD document 146
COLD document 62
COM object 455
com.venetica.vbr.client 389
common error
CFS-IS, menu item mismatch 127
common errors
CFS-IS, federation 126
CFS-IS, field size mismatch 127
common mistake
database related problems, CFS (using Content Integrator) 271
common mistakes
CFS (using Content Integrator) 271
CFS-CMOD 184
Common Viewer 307, 317
complex properties
creating 441
updating 442
complex property 437
Component Object Model (COM) 295
concurrency 455
config.bat 327, 330, 363
configuration
IBM Content Integrator 327
Configuration API 304, 317, 386
Configuration API p 426
configuration field 480
configuration server 309
configure security
CM8, CFS (using Content Integrator) 204
configuring and enabling
Enterprise Records 277
connector 47, 293, 305, 450, 455, 458, 462, 485
building jar file 483
building, deploying, and configuring 483
configuring 484
deploying 484
extending 485
planning and designing 450
connector configuration 321, 329–331, 334, 356
associate logging settings 334
Content Manager 343
EMC Documentum 336–338
FileNet Content Services 356
Hummingbird DM 340
IBM FileNet Content Manager 361, 371
Microsoft SharePoint 372, 374
RMI Proxy Connector 332
time zone context 334
connector development 295
connector implementation 294, 301
connector plug-in 494
configuring 496
creating 489
plan 490
connector plug-ins 318, 488
connector properties 331, 335–336
Connector Service Provider Interface (SPI) 294, 304, 386
connector.properties 481
ConnectorBridge 479, 487
ConnectorBridgeFactory class 487
ConnectorConfigFactory 481, 483
containability 37
content
definition 6–7, 37
deletion process 48
discover 34
federated 5, 13, 36, 39–40, 47, 49
federated, records management operations 13
federated, retention schedule 23
federated, source repository 47–48
unstructured 4
when to federate 21
when to integrate 21
content data 6
Content Engine (CE) 4–5, 37, 39, 43, 48

API 47
error log 121
trace log 121
Content Federation Service (CFS) 4, 14, 18–19
Content Federation Services (CFS)
architecture 34, 36, 38
functional components 41
Content Integrator
data model maps to the P8 190
Content Integrator Admin Tool 215
Content Integrator ItemClass 190
content item 7, 26, 339–340, 352, 360, 383
property names 371
content lockdown process 49
content management
Web content management 9
content management repositories 295
content management strategy 5
content management system 7, 10–11, 14, 499–500
basic architecture 7
federated 11–13
Content Manager connector
configuration 343
Content RepoItem 190
content repository 37
content retrieval components 46
content retrieval process 47
content streaming
native 319
create
CFS (using Content Integrator) 195
CFS operation 40
CFS-CMOD 149
CFS-IS operation 56
createItem(…) 478
createItem(…) method 475
createItemFromFiles(…) method 475
custom document class
create in CE 168
custom property 335, 338–340, 375, 481
name 359
UsageMode 356
USE_CHRONICLE_ID 339
valid uses 359
value 359
customer relationship management (CRM) 514

## D

data
structured 6
unstructured 6
data map 298, 305, 394
Content Integrator 192
Data Map Designer 319
data model 298
Image Services 52
mapping Content Integrator to P8 190
mapping, IS to P8 52
database
CFS-IS federation database 55
federation 43
federation database 45
federation database, CFS-CMOD 148
IS doctaba table 125
database indexes
CFS-IS 138
database related problems
CFS (using Content Integrator) 271
database table
doctaba 54
date and time properties
handling 407
date mismatch
common error, CFS-IS 127
declaration 25
delete
CFS (using Content Integrator) 197
CFS operation 41, 49
CFS-CMOD 151, 183
CFS-IS operation 58
deletion
CFS (using Content Integrator) 276
deletion process
content 48
design principle
IBM Content Integrator 314
destination repository 43
destruction 25
IS document 114
Developer and User Services layer 292–293
Developer and User Services layer, 303
digital capture 8
digital capture device 5
direct mode 316, 318
disaster recovery 26
discover content 34

Archived

disparate systems 10
disposition 29, 182
CFS-IS 114
doctaba database table 54
document
add IS document, testing 100
bidirectional deletion, CFS-IS 62
CFS-IS document class consolidation 137
CMOD 146
COLD 62
definition 37
existing IS document, enable federation 98
external document 37, 45, 47
external, version 45
federate document, CFS-CMOD 168
federate document, CMOD 180
federated 13, 35–37, 45, 47
federated IS document 56
federated, catalog entries 45
federated, check in 45
federated, definition 37
federated, intermediate form 38
federated, main difference 35
federated, property 40
federated, security 50
federated, source, relationship 35
federating new document 59
find IS document 101
hold 24–25
IS document 52, 55, 108
IS document destruction 114
IS document search 102
IS federated document, property update 57
new IS document, enable federation 97
non-federated 35–36, 40, 48
P8 52
retrieve federated document, CFS-IS 61
source 23, 35, 37–38
source, version 40
verify federated document, CFS (using Content Integrator) 255
view federated document, CFS-CMOD 176
document class 63
IS 52
P8 52
document class consolidation
CFS-IS 137
document life cycle 60
document management 7–8
document name 350
Document Parts 190
document version
federated, CFS (using Content Integrator) 196
domains 63

## E

ECM system 8
eDiscovery 24
efficiency 311
electronic discovery 24
EMC Documentum 329, 336–337, 506, 508–509
content management system 336, 509
docbase name 338
documentation 336
Foundation Class 336–338
Foundation Classes 337
logical mapping 339
EMC Documentum connecter
additional information 339
configuration 336–338
EMC Documentum Foundation Classes 336
EMC Documentum Foundation Classes (DFC) 336–338
encryption 301
encryption framework 302
Enhanced security model 434
enterprise content management (ECM) 4
Enterprise Record 355, 508–509
Enterprise Records 50, 508
configuring and enabling 277
Enterprise Records (previously known as IBM FileNet Records Manager) 14, 19, 21, 23
error log
Content Engine 121
event log
Image Services 121
execute
query 397
executeFinder 471
executeFinder() 469
executeFinder(ItemFinder) 468
executeQuery(Query) 471
Export 269
export
IS catalog 130
export tool
IS 70

Archived

export utility
CFS-CMOD 148
configure and run, CFS-CMOD 173
export utilityCFSOD 154
export_log table 55, 123
Exporter 37, 45
CFS (using Content Integrator) 193
CFS-IS 54
definition 43
exporter
CFS-CMOD 148
Exporter log
CFS (using Content Integrator) 263
Exporter performance
CFS (using Content Integrator) 281
exporter, run, CFS-CMOD 153
exporting
CFS (using Content Integrator) 194
CFS-CMOD 149
CFS-IS 55
External Content Reference 47
external document 37, 45, 47
new version 45
external repository 12, 34–37, 39–40
corresponding object 50
definition 37
fixed content device logs 48–49
interface 46
native application 50
source document 40
EXTERNAL_ATTACHMENT 451
EXTERNALIDENTITY table 154

## F

factory class 479
FedAdmin log 263
FedAdmin Rule Builder
tips 283
FedAdmin Rule Management
tips 283
federate and lockdown during load
CFS-CMOD 182
federate content
when 21
federate document
CFS-CMOD 168
federated business process management 28–29
federated content 5, 13, 36, 39–40, 47, 49, 509
advanced records management capabilities 5
records management operations 13
retention schedule 23
source repository 47–48
federated content management 4, 12
application 11
definition 5, 11
implementation strategies 15
overview 3
system 11
federated content management system 11–13
federated document 13, 35–37, 40, 45, 47
catalog entries 45
check in 45
CMOD, declare as record 180
definition 37
intermediate form 38
IS, property update 57
main difference 35
property 40
relationship to the source document 35
retrieve, CFS-IS 61
security 50
update, master catalog 43
verify, CFS (using Content Integrator) 255
view, CFS (using Content Integrator) 261
view, CFS-CMOD 176
federated document operations
CFS (using Content Integrator) 194
federated document version
CFS (using Content Integrator) 196
federated IS document 56
federated model 12
federated query 429
Federated Query Transformation 429
federated record 29
federated records management 13
CFS (using Content Integrator) 275
CFS-CMOD 179
CFS-IS 112
federated search 13, 24, 296–297
federated security 13
federating new document 59
federation 12
CM8 documents 200
enable existing IS document 98
enable new IS document 97
impact, CFS-IS 60
post federation migration, CFS-IS 69

Archived

records 22–23, 25
federation approach 14
federation components 42
federation database 43, 45
CFS-CMOD 148
CFS-IS 55
Content Integrator 193
federation implementation strategy 34
federation process 45
CFS-CMOD 148
federation rule
scheduling 257
view status, CFS (using Content Integrator) 267
federation rules
create, CFS (using Content Integrator) 247
Federation Services layer 292–293, 296, 300
federation strategy 15–16
field
CMOD 146
field size mismatch
common error, CFS-IS 127
file system connector 461
FileNet Content Services connecter
configuration 356
FileNet CS connecter
additional information 360
find IS document
testing CFS-IS 101
fixed content device 46, 48–50
fixed storage area 47
folder 37, 452, 465–466
folder retrieval 463
full-text search 371, 510
property-based searches 376

## G

getContent() 401
getFolder() 401
getFolder(String) 465
getRepoItem() 401

## H

helper method 464
heterogeneous repositories 11
High Performance Image Import (HPII) 20
highly available environment
CFS (using Content Integrator) 284
hold
document 24–25
holds
CFS-CMOD 182
Hummingbird DM connector
configuration 340

## I

IBM Content Federation Services 34
IBM Content Integrator 4–5, 18, 20, 41, 49, 304, 321–322, 324, 499–502, 514–515
advanced and enhanced security models 360
deployment architecture 516
design philosophies 501
Direct use 499, 501
Embedded use 509
enhanced security model 351
enterprise deployments 329
following features 507
important consideration 515
installation 322
multi-part content 340
read-token security models 339, 371, 382
sizing guidance 513
top-level folders 377
IBM Content Integrator Administration Tool 394
IBM Content Manager xiv–xv, 5, 19–21, 41, 83, 145, 151, 155, 189, 192, 202, 295, 315, 329, 343–345, 427, 431, 438, 504, 508, 510
annotation type 352
connecter 352, 355
connector configuration 344
data 347
data store 347
machine 346
OnDemand 21
repository 353, 355
repository item name 350
server 346
system administration tool 353
work list 354
work node 354
workflow 354
IBM Content Manager OnDemand 508
IBM FileNet
application 19
Business Process Manager 8, 19–20, 36
Capture 20
Content Federation Service 356

Archived

Content Manager 5, 8, 18–19, 329, 362, 365, 504, 508, 514
Content Manager Application Engine 365
Content Manager connector
configuration 361
Content Manager display name 370
Content Manager domain name 371
Content Manager engine 370
Content Manager installation directory 362, 365
Content Manager server 361, 365–366
Content Manager system 365, 368
content repository 24
Content Service 5, 20–21
Content Service (CS) 329, 360
CS content management system 358
CS document version list 361
IDM 356, 360
IDM Desktop 357
Image Services repository 19
P8 5, 13, 19, 504, 509
P8 application 20
P8 Platform 21
Records Manager 36, 355
repository 371
IBM FileNet Content Manager
Application Engine 362, 365
connecter 371
display name 370
domain name 371
engine 369–370
installation 362
installation directory 362, 365
item class 371
JNDI 363, 366
object store 369
property 370
security 371
server 361–362, 365
symbolic name 370
system 366, 368, 504
transaction logging 369
WcmApiConfig.prop erties 367
WcmApiConfig.properties 364
Workplace Application 370
IBM FileNet CS
content management system 360
document version list 361
system 359–360
IBM FileNet IDM
Desktop 356
Web client 356
IBM ImagePlus 504
system 504
IBM Records Federation Services 22
ICMBase 190
Image Services
event log (elog) 121
Image Services data model 52
impact
federation, CFS-IS 60
implementation strategies
federated content management 15
implementation strategy
federation 15–16, 34
integration 15–16
implementing CFS-IS
planning 58
import
improve performance, CFS-IS 130
import performance
CFS-CMOD 154
import queue 38, 43
import request 43, 45
definition 38
Importer 45
CMOD 148
importer 38
Content Integrator 193
definition 43
Importer performance
CFS (using Content Integrator) 283
importing
bulk importing, CFS-IS 130
CFS (using Content Integrator) 194
CFS-CMOD 149
CFS-IS 56
Index Service
IS 54
industry solution
content access and federated search platform 501
Document exchange platform 506
Federated records solution 507
information silo 5, 9, 11
InfoSphere Enterprise Records 508
infrastructure support 27
installation and configuration
CFS (using Content Integrator) 202

Archived

CFS-CMOD 154
Integrate API 304
integrate content
when 21
integration services 26
Integration Services layer 293
components 292
integration strategy 15–16
INTERNAL_ATTACHMENT 451
IP address 333, 342, 346
IS
records management 63
IS catalog export 130
IS database table doctaba 125
IS document 52, 55, 108
destruction 114
existing, enable federation 98
federated 56
find 101
new, enable federation 97
search 102
IS document class 52
IS export tool 70
IS federated document
property update 57
IS Index Service 54
IS log files 121
Item
CM8 Item 190
item attribute 466
item ID 478
item properties 467
Item Type
CM8 Item Type 190
ItemClass 190
ItemFinders 453, 468–469
items 44

## J

Java Runtime Environment (JRE) 344, 363, 367
Java Virtual Machine (JVM) 334, 337, 342, 455
JDBC Drivers
CFS (using Content Integrator) 272
JVM instance 310

## L

LIKE operator 474
load
document, CMOD 146
federate and lockdown, CFS-CMOD 182
lockdown 50
CFS (using Content Integrator) 197, 275
CFS operation 41
CFS-CMOD 150, 181
CFS-IS operation 58, 112
definition 23
during load, CFS-CMOD 182
lockdown process
content 49
LockDownPrincipal custom property 340, 360, 382
lockDownRecord 340, 355, 360
log
Content Engine error log 121
Content Engine trace log 121
Exporter log, CFS (using Content Integrator) 263
FedAdmin, CFS (using Content Integrator) 263
Image Services event log 121
recovery log 128
trace log collecting, CFS-CMOD 185
trace log in CE 185
log files
IS and CE 121
logging 305
Logging server 309
logon encryption framework 302
logs
CFS (using Content Integrator) 262
fixed content device 48–49

## M

Magnetic Storage and Retrieval (MSAR) 60
manifest file 495
MANIFEST.MF 483
mapping 452
IS data model to P8 52
mapping repository 457
marking set 140
create 141
master catalog 17, 19, 34–35, 39–40
definition 34
master catalogue
catalog entries 42
update federated document 43
maximum result 471
MaxUserPort 70

Archived

memory 475
menu item mismatch
common error, CFS-IS 127
metadata 38, 453
attributes 44
change, CFS-IS 61
definition 37
metadata retrieval 409
metadata schema 298
Microsoft SharePoint
connecter 372, 374–375
connector-specific property 374
Service 372
migration
post federation migration, CFS-IS 69
MIME type 178, 195, 302, 348, 350, 359
mapping image/tiff 375
mappings 375
MODCA document
federating and viewing 262
model
federated 12
multipart content 451
MultiQuery 397
enable thread pooling 398
MultiQuery implementation 398

## N

native content retrieval 410
native content streaming 319
Near Line Storage (NLS) 60
node Name 346
non-English character 352
non-federated document 35–36, 40, 48

## O

object store schema 35
ontent and metadata 450
OpenText Livelink 329, 380–381, 506, 510
OpenText Livelink connecter
additional information 382
configuration 380
operational capability 25, 27
Optical Storage and Retrieval (OSAR) 60
orICMTextBase 190
out of memory error 475

## P

P8 document 52, 190
P8 document class 52, 190
P8 document class, CFS-CMOD 146
pages 190, 451
paperless statement 505
online statement service 505
performance 25, 409
Exporter, CFS (using Content Integrator) 281
import performance, CFS-CMOD 154
improve import, CFS-IS 130
planning
CFS (using Content Integrator) 197
CFS-CMOD 151
CFS-IS 63
CFS-IS capacity planning 136
CFS-IS, capacity 136
implementing, CFS-IS 58
planning consideration
CFS (using Content Integrator) 199
Portal Document Manager (PDM) 510
post federation migration
CFS-IS 69
preservation 25–26
preservation feature 25
process
business process 4
business process management 3, 9, 12
properties 190
property 52
property index 133
property retrieval 467
property update
IS federated document 57

## Q

query
Content Integrator 192
execute 397
federated query 429
query expression 472–473
query object
create 395
QueryResults 471
queue
import queue 38, 43

Archived

## R


read
  CFS operation 40
Read Token security model 436
record
  federated 29
record declaration 37
records control 22
records declaration
  CFS-CMOD 179
records disposition schedule 23
records federation 22–23, 25
records management 8
  CFS (using Content Integrator), best practices 287
  federated 13
  federated, CFS-CMOD 179
  federated, CFS-IS 112
  IS 63
records management capabilities 5
records management operations
  federated content 13
recovery log 128
Redbooks Web site 524
  Contact us xvii
remote caching 134
Remote Method Invocation (RMI) 514, 517
Remote Method Invocation (RMI) Proxy Connector 296
Remote Method Invocation (RMI) Proxy Connector Server 387
Remote or Distributed Session Pool Server. 415
remote repositories 34
RepoBrowser 468
RepoItemHandle 400, 466
repository 452–453, 458
  content 37
  definition 10
  destination 43
  external 12, 34–37, 39–40
  external, corresponding object 50
  external, definition 37
  external, fixed content device logs 48–49
  external, interface 46
  external, native application 50
  external, source document 40
  federated 35
  remote 34
  source 23, 29, 38, 40–42, 48
  source, document types 44
  source, federated content 47–48
  target 43
  virtual 11
repository profile 293, 474
ResultRows 398
results
  aggregate 34
retention period 114
retention schedule 23
retrieval keys
  CFS-IS 138
retrieve
  CFS (using Content Integrator) 195
  CFS-CMOD 150
  CFS-IS operation 57
retrieve federated document, CFS-IS 61
retrieve metadata, 401
retrieve the native content 402
retrieving item 463
review 25
review process 29
RMI Proxy Connecter 324, 329–330, 332, 514, 517–518
RMI Proxy Connector 208, 329
RMI proxy connector 280
RMI Proxy Connector server 309–310
  environment 330
  use 331
root folder 465
run exporter 153
run_soa_sample 448


## S


sample client application 391
scalability 25
Scheduler 45
scheduling utility, CFS-CMOD 148
Schema Map 44
schema map
  CFS-IS 53
  Content Integrator 191
Schema Mapper 43
schema mapper
  CFS-CMOD 148
  Content Integrator 194
schema mapping 18, 20
schema mapping editor


Archived

CFS-IS 53
search
federated 13, 24
search IS document
testing CFS-IS 102
search request, 298
Secure Sockets Layer (SSL) 378
security
configure security for CM8, CFS (using Content Integrator) 204
federated 13
federated document 50
security component 300
security mapping
annotation, configure 105
security model 301, 318, 432
security models 430
security principal 37
CM8 200
selection criteria 395, 402
selection properties 395, 402, 469
Service Oriented Architecture (SOA) 18
session management 456
session model 299
Session Pool 412, 420
session pool 299, 305, 312, 394
distributed 312
Session Pool Server 416
local 415
Session Pool Servers
local and remote 415
session pools
local 300
silo effect 9
Simple Object Access Protocol (SOAP) 305
Simple Security model 432
Single Document Storage (ISDS) 61
single interface 510
single sign-on 300, 313
Single sign-on (SSO) 349
Single Sign-On reference implementation 319
Single Sign-On system 424
single sign-on system 300
Single Sign-On, 421
SOA Web services 318
source document 23, 35, 37–38
new version 40
relationship to the federated document 35
source repository 23, 29, 38, 40–42, 48
definition 37
document types 44
federated content 47–48
SPI 408
SSO implementation 425
strategy
content management 5
structured data 6
Swing-based graphical user interface 389

## T

target repository 43
TcpNumConnections 70
TcpTimedWaitDelay 70
testing
CFS-IS configuration 100
testing CFS-IS
add IS document 100
threadsafe 455
throughput 311
time zone 407
trace log
Content Engine 121
trace logging
enabling in CE 185
transfer 25
trigger 29
troubleshooting
CFS (using Content Integrator) 262
CFS-CMOD 184
CFS-IS 121
collecting trace logs 185
tuning
CFS-IS 130

## U

unintentional disclosure 28
Unique identifier 454
UNIQUE_ID 454
unsealing bundle 461
unstructured content 4
unstructured data 6
update
CFS (using Content Integrator) 195
CFS operation 40
CFS-CMOD 150
CFS-IS operation 57
user credential 300

Archived

## V

vbr.as.direct.rmi.urls 394
verifty federated document 255
version 37, 40
view federated document
    CFS-CMOD 176
view services component 302
view services framework 305
virtual repository 11

## W

WcmApiConfig.properties
    file 366
    resource bundle 368
Web Application Archives (WAR) 447
Web content management 9
Web content mangement 9
Web crawling 294
Web search engine 34–35
Web services 295, 305
Web Services API (WS-API) 443
Web Services API (WSAPI) 305
Web Services Description Language (WSDL) 444
wildcards 474
Work item 351, 353
WORK_QUEUE 463
workflow 29
WorkplaceXT 176
WorkQueue 468
WS-API J 448

## X

XML 443

## Y

y 402

Archived

Archived

Redbooks

# Federated Content Management with IBM Solutions

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

Archived

Archived

Archived

# Federated Content Management

## Accessing Content from Disparate Repositories with IBM Content Federation Services and IBM Content Integrator

**Learn about federated content management**

**Install and configure Content Federation Services**

**Use Content Integrator connectors and client APIs**

Today, businesses have valuable operations data spread across multiple content management systems. To help discover, manage, and deliver this content, IBM provides IBM Content Federation Services and IBM Content Integrator. This IBM Redbooks publication introduces the concept of federated content management and describes the installation, configuration, and implementation of these product offerings.

*IBM Content Federation Services*, available through IBM FileNet Content Manager, is a suite of three federated content management services based on the federation implementation strategy. We describe how to install and configure Content Federation Services for Image Services, Content Manager OnDemand, and IBM Content Integrator.

Using an integration implementation strategy, *IBM Content Integrator* provides a repository neutral API that allows bidirectional, real-time access to a multitude of disparate content management system installations. We present connector configuration details to frequently encountered content management systems. We provide detailed instruction and sample implementations using the product's Java and Web Services APIs to access content stored in repository systems. This book is intended for IT architects and specialists interested in understanding federated content management and is a hands-on technical guide for IT specialists to configure and implement federated content management solutions.

SG24-7742-00 ISBN 0738433977

**INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

**BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**
**ibm.com**/redbooks