



IBM FileNet P8 Platform and Architecture

Architecture and expansion products

Enterprise content management

Scalability and distribution



Wei-Dong Zhu
Kameron Cole
Adam Fowler
Michael Kirchner
Bruce J McDowell
Chuck Snow
Mike Winter
Margaret Worel

ibm.com/redbooks

Redbooks



International Technical Support Organization

IBM FileNet P8 Platform and Architecture

July 2009

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

First Edition (July 2009)

This edition applies IBM FileNet P8 Release 4.5

© Copyright International Business Machines Corporation 2009. 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

Figures	xi
Notices	xv
Trademarks	xvi
Preface	xvii
The team that wrote this book	xvii
Become a published author	xx
Comments welcome	xx
Chapter 1. IBM FileNet P8 Platform overview	1
1.1 Today's challenge	2
1.1.1 The problem defined	2
1.2 Enterprise content management and its capabilities	4
1.2.1 Enterprise content management capabilities	5
1.3 IBM FileNet P8 Platform	6
1.3.1 IBM FileNet P8 solutions by industry	8
1.3.2 Core IBM FileNet P8 products	9
1.3.3 IBM FileNet P8-driven expansion products and applications	10
1.3.4 IBM FileNet P8 architecture and its core engines	17
1.3.5 Support for Java, .NET, and XML Web Services Frameworks	19
1.3.6 Content federation	19
1.3.7 Performance and scalability	22
1.4 Enterprise Reference Architecture	23
1.5 Summary	26
Chapter 2. Core component architecture	27
2.1 IBM FileNet P8 Platform overview	28
2.2 Content Engine	29
2.2.1 Data model	31
2.2.2 Authentication	32
2.2.3 Authorization	32
2.2.4 Event framework	33
2.2.5 Life cycles	34
2.2.6 Storage services	34
2.2.7 Full-text indexing	36
2.2.8 Publishing	38
2.2.9 Classification	38
2.2.10 Protocols	39

2.2.11 Tools	39
2.3 Process Engine	40
2.3.1 Data model	41
2.3.2 Security	44
2.3.3 Process orchestration	44
2.3.4 Event logging	45
2.3.5 Analytics	46
2.3.6 Monitoring	46
2.3.7 Rules framework	46
2.3.8 Protocols	46
2.3.9 Tools	47
2.4 Application Engine	47
2.4.1 User preferences	49
2.4.2 Microsoft Office and Outlook integration	49
2.4.3 Component Manager	50
2.5 IBM FileNet Records Manager	50
2.5.1 Data model	51
2.5.2 File plans	52
2.5.3 Additional references	54
Chapter 3. Expansion products for content ingestion	55
3.1 Expansion product overview	56
3.2 Content ingestion products overview	57
3.3 IBM Content Collector	58
3.3.1 IBM Content Collector overview	58
3.3.2 System architecture	61
3.3.3 Connection and integration points	65
3.3.4 ICC summary	66
3.4 IBM FileNet Capture	67
3.4.1 Capture process overview	67
3.4.2 Capture systems architecture	69
3.4.3 IBM FileNet Capture products overview	70
3.4.4 IBM FileNet Capture Professional	72
3.4.5 IBM FileNet Capture Advanced Document Recognition (ADR)	75
3.4.6 IBM FileNet Remote Capture	78
3.4.7 IBM FileNet Fax/MFP	78
3.4.8 Integration points	79
3.4.9 Capture summary	79
3.5 Summary	80
Chapter 4. Expansion products for connectors and federation	81
4.1 Connectors and federation products overview	82
4.2 IBM FileNet Application Connector for SAP R/3	82

4.2.1 IBM FileNet Application Connector for SAP (ACSAP) R/3	83
4.2.2 Data and document archiving	83
4.2.3 Application Connector for SAP Enterprise Portal (ACSAP EP)	84
4.2.4 IBM SAP connectors architecture	85
4.2.5 ACSAP R/3-J2EE architecture	86
4.2.6 ACSAP EP architecture	86
4.2.7 SAP summary	87
4.3 IBM Content Integrator	88
4.3.1 Content Integrator architecture	88
4.3.2 Content Integrator summary	90
4.4 Content Management Interoperability Services	91
4.4.1 Integration and connection points	91
4.4.2 CMIS summary	93
4.5 IBM FileNet Services for Lotus Quickr	93
4.5.1 Integration points	94
4.5.2 Architecture	97
4.5.3 Quickr summary	100
4.6 IBM FileNet Connectors for Microsoft SharePoint	100
4.6.1 IBM FileNet Connector for SharePoint Document Libraries	101
4.6.2 Architecture	101
4.6.3 Services and integration points	103
4.6.4 IBM FileNet Connector for SharePoint Web Parts	104
4.6.5 Summary of IBM FileNet Connectors for SharePoint	105
4.7 Summary	106
Chapter 5. Expansion products for application framework	107
5.1 Application framework products overview	108
5.2 Electronic forms (eForms)	109
5.2.1 Architecture	110
5.2.2 Integration and protocols	112
5.2.3 Customization and integration with third-party applications	113
5.2.4 eForm summary	114
5.3 Business Process Framework	114
5.3.1 Architecture	122
5.3.2 BPF summary	124
5.4 Business Activity Monitor and Cognos Now	124
5.4.1 Architecture	126
5.4.2 BAM summary	128
5.5 Summary	128
Chapter 6. Expansion products for search, classification, and discovery .	129
6.1 Search, classification, and discovery product overview	130

6.2 IBM Classification Module	130
6.2.1 Integration and connection points	132
6.2.2 Architecture	132
6.2.3 ICM summary	133
6.3 eDiscovery Manager and Analyzer	134
6.3.1 Architecture	134
6.3.2 Integration and connection points	135
6.3.3 Summary of eDiscovery	136
6.4 IBM OmniFind Enterprise Edition	137
6.4.1 Integration and connection points	137
6.4.2 Architecture	137
6.4.3 OmniFind Enterprise Edition summary	138
6.5 Summary	139
Chapter 7. Enterprise content management	141
7.1 Anatomy of an ECM infrastructure	142
7.1.1 Store and retrieve	143
7.1.2 The case for metadata	144
7.1.3 Enterprise catalog and content federation	145
7.1.4 Security and access control	147
7.1.5 Object classes and inheritance	148
7.1.6 Processes	154
7.2 Content event processing	154
7.2.1 Active content	154
7.2.2 System events	155
7.2.3 Custom events	158
7.2.4 Custom event actions	159
7.2.5 Classification and taxonomies	161
7.3 Content life cycle	163
7.3.1 Document life cycle	163
7.3.2 Life cycle and content storage	164
7.4 Business processes	166
7.4.1 Content-centric BPM	166
7.4.2 Complex interactions with external systems	170
7.4.3 Business Rules Engines	173
7.4.4 Auditing and monitoring	174
7.4.5 Analysis and optimization	178
7.5 Records management	180
7.5.1 Basic requirements	181
7.5.2 Product features and leveraged IBM FileNet P8 Platform capabilities	181
7.5.3 Platform extensions	185
7.6 Summary	185

Chapter 8. Security	187
8.1 Authentication and authorization	188
8.1.1 Authentication in IBM FileNet P8	188
8.1.2 Single Sign-On	189
8.2 Securing IBM FileNet P8 core components	192
8.2.1 Content Engine	192
8.2.2 Process Engine	194
8.2.3 Workplace and WorkplaceXT access roles	195
8.3 Access to information	196
8.3.1 Document security	196
8.3.2 Default instance security	199
8.3.3 Object owner permissions	200
8.3.4 Security precedence and inheritance	201
8.3.5 How authorization is calculated	202
8.3.6 Authorization calculation example	203
8.4 Setting security across the enterprise	205
8.4.1 Marking sets	205
8.4.2 Security policies	209
8.4.3 Document life cycle policies	211
8.4.4 Ethical wall	217
8.4.5 Creating a shared service	218
8.5 Security requirement changes with time	219
8.5.1 How document and process life cycle affects security	219
8.5.2 Managing security updates	220
8.5.3 Update using business processes	223
8.5.4 Critical records declaration, holds, and disposition	224
8.5.5 Institutional reorganizations	226
8.6 Content-level security	227
8.6.1 Local copies on users' machines, client cache files	228
8.7 Network security	229
8.7.1 Demilitarized Zones (DMZ)	229
8.7.2 Encryption on the wire	230
8.7.3 Web services security	231
8.8 Reporting and auditing	233
8.8.1 Logging into the Content Engine	234
8.8.2 Logging into the Process Engine	235
8.8.3 Process Analyzer	237
8.8.4 Content Engine object-level JDBC provider	238
8.8.5 Case Objects as historical records	239
8.9 A practical example: Re-insurance placement and litigation	240
8.10 Summary	247
Chapter 9. Scalability and distribution	249

9.1 Overview	250
9.1.1 Horizontal scaling: scale out	251
9.1.2 Vertical scaling: scale up	251
9.1.3 Virtualization	253
9.1.4 Load balancing	255
9.2 Scaling the IBM FileNet P8 core engines	257
9.2.1 Application Engine	257
9.2.2 Content Engine	262
9.2.3 Content Search Engine	272
9.2.4 Process Engine	275
9.2.5 Summary	279
9.2.6 Scaling add-on products	280
9.2.7 IBM FileNet Image Services	289
9.3 Tuning the IBM FileNet P8 Platform for performance	291
9.3.1 J2EE Application Server	292
9.3.2 Database	293
9.3.3 Application design	294
9.4 Distributing an IBM FileNet P8 system	295
9.4.1 Geographically dispersed users	296
9.4.2 Disaster recovery	302
9.5 IBM FileNet P8 in a DMZ environment	303
9.6 Sample deployment	306
 Chapter 10. Architecting an IBM FileNet P8 solution	 311
10.1 Solution overview	312
10.2 Solution template: Customer Services Support	313
10.2.1 Scenario	313
10.2.2 Business problems and their solutions	313
10.2.3 Customer architectural constraints	315
10.2.4 Solution architecture	316
10.2.5 Solution processes	317
10.2.6 Future enhancements	318
10.3 Solution template: Enterprise-wide Document Management	319
10.3.1 Scenario	320
10.3.2 Business problems and their solutions	320
10.3.3 Customer architectural constraints	326
10.3.4 Solution architecture	327
 Related publications	 331
IBM Redbooks	331
Online resources	331
How to get Redbooks	332
Help from IBM	332

Index 333

Figures

1-1	Core IBM FileNet P8 engines	17
1-2	Content Integration services for IBM FileNet P8	20
1-3	Layer 1 of the IBM FileNet P8 Enterprise Reference Architecture	25
2-1	IBM FileNet P8 core components	29
2-2	Content Engine internal system architecture	30
2-3	Content Engine internal database structure	31
2-4	Content Engine storage options	36
2-5	Content Search Engine system architecture	38
2-6	The Process Engine system architecture	41
2-7	Process Engine database	42
2-8	Process Engine database schema	43
2-9	Process orchestration	44
2-10	Process orchestration interaction	45
2-11	Application Engine system architecture	49
2-12	IBM FileNet Records Manager architecture as an extension of the IBM FileNet P8 Platform	51
2-13	File plan structure	53
3-1	IBM Content Collector Configuration Manager with a sample task route	59
3-2	ICC e-mail search Web application	61
3-3	IBM Content Collector system architecture	63
3-4	IBM Content Collector Archive Engine	64
3-5	ICC toolbar in Lotus Notes	66
3-6	Basic capture functionality	68
3-7	Advanced capture process	69
3-8	IBM FileNet Capture in a distributed architecture	70
3-9	ADR capture process	76
4-1	High level IBM FileNet P8-SAP architecture	85
4-2	ACSAP R/3 J2EE architecture	86
4-3	ACSAP Enterprise Portal: Knowledge Management architecture	87
4-4	Content Integration architecture	89
4-5	Prototype of the interface between the IBM FileNet repository and SAP	92
4-6	The Document Picker dialog box	95
4-7	Lotus Quickr connectors integrate seamlessly	97
4-8	Lotus Quickr modules	98
4-9	Services for Lotus Quickr connect using the Lotus Quickr connectors	99
4-10	Quickr REST connections	100
4-11	IBM FileNet Connector for SharePoint document libraries architecture	102
4-12	The core Windows services	103

5-1	The FileNet eForms Designer application	110
5-2	eForms architecture as supported by the IBM FileNet P8 Platform	111
5-3	A wizard-driven form created using the eForm JavaScript APIs	114
5-4	Sample user window with different user roles and inbaskets	116
5-5	BPF Explorer with Inbasket properties	117
5-6	Sample generic case application in BPF.	119
5-7	Case management interface	120
5-8	BPF architecture	123
5-9	BAM interaction with Process Analyzer	126
5-10	BAM architecture.	127
6-1	ICM learning data flow	132
6-2	ICM server architecture.	133
6-3	High level look at eDiscovery Manager.	135
6-4	eDiscovery Manager as integrated with IBM FileNet P8.	136
6-5	OmniFind Enterprise Edition communication	138
7-1	Document class hierarchy sample	149
7-2	ECM and SOA.	173
7-3	Monitoring dashboard visualized by BAM.	176
7-4	Common Base Events for Process Engine	178
7-5	FileNet Process Analyzer data flow	179
7-6	Separation of document objects and record objects	182
8-1	Typical Single sign-on (SSO) authentication behavior	191
8-2	Default instance security and default owner settings on a document class . 199	
8-3	The marking set properties and marking properties dialog boxes	207
8-4	Use permission propagated in security classification marking set example . 208	
8-5	Assigning a default security policy to a new instance of a class	210
8-6	A simple document editing, approval and publishing life cycle	212
8-7	Setting the class of the object this class property is to inherit from. . . .	214
8-8	Example of searching for a delegate class	216
8-9	File tracking options to ensure local copies are deleted	229
8-10	Providing WS-Security credentials for a Web service.	232
8-11	Configuring incoming authentication for a Web services call	233
8-12	Content Engine logging configuration	235
8-13	The default Process Engine log with extra configured user-defined fields . 236	
8-14	Event log configuration in the Process Engine	237
8-15	Logged in as Administrator, we can see the confidential properties . .	243
8-16	Logged in as a user who cannot see the confidential information . . .	244
9-1	Combination of vertical and horizontal scaling for J2EE applications . .	252
9-2	Logical architecture of a virtualized environment	254
9-3	Hardware load balancing for Application Engine Web application farm .	259

9-4	Load distribution for Application Engine Web application using an http plug-in.	260
9-5	Configuration where applications use a particular Content Engine server	264
9-6	: Application server cluster based load balancing for Content Engine . .	266
9-7	Hardware load balancer fronting a Content Engine farm	268
9-8	Scaling out Content Search Engine indexing	273
9-9	Scaling out Content Search Engine retrievals.	275
9-10	Process Engine system with server farm	277
9-11	Process Engine system with standalone servers	279
9-12	Scaling ACSAP for R/3 with IBM FileNet P8 Content Engine	283
9-13	Scaling ACSAP for R/3 with Image Services	284
9-14	Horizontal scaling for IBM Content Collector	287
9-15	Image Services architecture	290
9-16	Central IBM FileNet P8 system	297
9-17	IBM FileNet P8 system with remote Application and Content Engine. .	298
9-18	Distributed Process Engines	301
9-19	DMZ with dual firewalls	304
9-20	IBM FileNet P8 DMZ deployment best practice	305
9-21	Application Engine in the DMZ	306
9-22	Design pattern for a clonable IBM FileNet P8 domain	308
9-23	Virtualization used to implement high availability on a shared infrastructure	309
10-1	Core solution architecture	316
10-2	Works Required process.	318
10-3	Solution with all optional components included.	319
10-4	Large site system architecture	328

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.

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 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®	iNotes®	Redbooks (logo)  ®
Cognos®	Lotus Notes®	Sametime®
DB2®	Lotus®	Symphony™
developerWorks®	Notes®	System x®
Domino®	OmniFind®	Tivoli®
FileNet®	Quickr™	WebSphere®
IBM®	Redbooks®	

The following terms are trademarks of other companies:

Cognos, and the Cognos logo are trademarks or registered trademarks of Cognos Incorporated, an IBM Company, in the United States and/or other countries.

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

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

Novell, the Novell logo, and the N logo are registered trademarks of Novell, Inc. in the United States and other countries.

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.

mySAP, SAP ArchiveLink, SAP NetWeaver, SAP R/3 Enterprise, SAP R/3, SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

EJB, Image Viewer, J2EE, Java, JavaScript, JDBC, JRE, JSP, JVM, Solaris, Sun, Sun Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Active Directory, Excel, Internet Explorer, Microsoft, MS, Outlook, SharePoint, SQL Server, Visio, 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.

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

Preface

IBM® FileNet® P8 Platform is a next-generation, unified enterprise foundation for the integrated IBM FileNet P8 products. It combines the enterprise content management with comprehensive business process management and compliance capabilities. IBM FileNet P8 addresses the most demanding compliance, content, and process management needs for your entire organization. It is a key element in creating an agile, adaptable enterprise content management (ECM) environment necessary to support a dynamic organization that must respond quickly to change.

In this IBM Redbooks® publication, we provide an overview of IBM FileNet P8 and describe the core component architecture. We also introduce major expansion products that extend IBM FileNet P8 functionality in the areas of content ingestion, content accessing through connectors and federation, the application framework, and discovery and compliance.

IBM FileNet P8 Platform supports enterprise content management. In this book, we discuss the anatomy of an ECM infrastructure, content event processing, content life cycle, and business processes.

Each IBM FileNet P8 product has its own functionality, but they are all built on top of the IBM FileNet P8 Platform. The support for security around authentication and access control of processes and content of these products is provided by the core platform. In this book, we describe the security issues to consider in an enterprise environment, how IBM FileNet P8 addresses them, and how to manage security effectively in an IBM FileNet P8 environment.

Another important topic that we address in this book is how the unique ability of the IBM FileNet P8 Platform can scale horizontally and vertically to respond to increasing load demands. We discuss the available options for each of the core platform engines and for the expansion products.

We designed this book to give IT architects, IT specialists, and IT Technical Sales a solid understanding of IBM FileNet P8 Platform, its architecture, its functions and extensibility, and its unlimited capabilities.

The team that wrote this book

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

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

Kameron Cole is a Senior IT Specialist and Managing Consultant for the IBM Center of Excellence, Content Management and Discovery. Kameron is a Sun™-certified J2EE™ Developer, IBM-certified WebSphere® Administrator, WebSphere Portal Administrator/Developer, and Enterprise Developer. His area of expertise includes architecting complex solutions for enterprise content management, content analysis, and discovery areas. He has a Master of Arts degree in Linguistics and a Bachelor of Arts degree in Computer Science, specializing in compiler construction and natural language processing. Kameron has written several technical books and developerWorks® articles in his specialty areas.

Adam Fowler is an Enterprise Content Management Technical Seller for the United Kingdom (UK) and Ireland. He has six years of experience in enterprise content management, business process management, and enterprise portal and systems integration. He has a degree in Computer Science from Aberystwyth University, Wales, UK. His areas of expertise include FileNet P8 4.x architecture and solutions, information about demand, and J2EE application design and development. Adam is an IBM FileNet Certified Professional in the P8 Platform, architecture and product set. Adam joined FileNet in 2006 and came into IBM in 2007. He has written extensively on approaches to supporting Technical Sales within IBM and has contributed previously to the IBM FileNet user forums, Tomcat user forums, and the Tomcat Book Project.

Michael Kirchner is a Lead IT specialist for IBM Enterprise Content Management Technical Sales in Germany. He has 10 years of experience in the enterprise content management and business process management marketplace. Michael has a Master of Electrical Engineering from the Ruhr-University in Bochum (Germany) and joined IBM in 2007 with nine years of FileNet experience, where he worked in the Professional Services organization as Team Lead, Project Manager, and Senior Systems Consultant. During this time, he was responsible for many implementations of ECM/BPM solutions based on IBM FileNet products. Michael has an expertise in IBM FileNet P8 4.0 architecture and is Technical Sales Certified for the FileNet P8 Content and Process products. He regularly advises clients in the Financial Services and Public/Distribution Sectors regarding their strategies in ECM.

Bruce J Mcdowell is a Senior IT Specialist for IBM Enterprise Content Management supporting the New York/New Jersey Metro Sales team in the US. He has 18 years of experience in the content management area. He has a degree in Sociology and Economics from Mount Union College and an AIIM ECM Practitioner designation. His areas of expertise include system and configuration analysis and design. Bruce has performed in a variety of roles in the content management arena, as a customer in insurance and banking, as a system integrator, and providing professional services for ECM software vendors.

Chuck Snow is an IT Specialist for the IBM Enterprise Content Management Software Development Group. He has more than 10 years of experience selling and supporting complex enterprise solutions within the northeastern United States. Chuck has a Bachelor's degree from Temple University in International Business and an MBA from Clark University in Management Information Systems and Global Business. He is involved with many fields within information management, with specialization in records management, collaboration, and knowledge management solutions.

Mike Winter is a Senior Technical Staff Member (STSM) and Enterprise Content Management architect responsible for architecture of the IBM FileNet Content Manager. Mike has more than 20 years of software development experience and has been heavily involved in the development of business process management and content management products within the FileNet brand for the past 15 years. He joined FileNet in 1993 and IBM in 2006 through a merger.

Margaret Worel is a Senior Systems Specialist Engineer (ITS), supporting demonstrations and rapid sales response for the IBM Enterprise Content Management Demo group in the US. She joined IBM through a merger with FileNet in 2006 and is currently focusing on demonstration best practices. Maig is a graduate of UCLA with over 12 years of software experience. She manages, designs, develops, and implements programs and procedures for data and process management.

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

Al Brown
Jon Brunn
Chuck Fay
Genifer Graff
Ulrich Leuthner
Tim Morgan
Joseph Raby
René Schimmer
Shawn Waters
IBM Software Group, US

Martin Pepper
IBM Software Group, UK

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

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 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



IBM FileNet P8 Platform overview

IBM FileNet P8 Platform is a next-generation, unified enterprise foundation for the integrated IBM FileNet P8 products. It combines the enterprise content management, comprehensive business process management, and extensive compliance capabilities to address a wide range of content-related business requirements. IBM FileNet P8 Platform is the key element in creating an agile, adaptable enterprise content management environment that is necessary to support a dynamic organization that must respond quickly to change.

In this chapter, we describe the issues that surround the exponential growth of electronic content and how IBM FileNet P8 Platform and its core products address these challenges. We also introduce a number of IBM FileNet P8 products that take advantage of these enterprise capabilities to further expand on the value proposition that IBM FileNet P8 has to offer.

The topics that we cover are:

- ▶ 1.1, “Today’s challenge” on page 2
- ▶ 1.2, “Enterprise content management and its capabilities” on page 4
- ▶ 1.3, “IBM FileNet P8 Platform” on page 6
- ▶ 1.4, “Enterprise Reference Architecture” on page 23
- ▶ 1.5, “Summary” on page 26

1.1 Today's challenge

The rate at which the volume of unstructured content has grown is staggering. Countless file formats, such as Web pages, spreadsheets, emails, scanned images, word processing documents, presentations, and multimedia, now comprise the bulk of public and private sector intellectual assets, driving essential business processes and enabling internal and external information sharing. With the proliferation of all of these forms of information, however, comes a set of challenges.

Left unchecked, the terabytes or even petabytes of content produced in just a year can pose significant obstacles to productivity, which unnecessarily escalates storage costs, and exposes an organization to regulatory, competitive and operational risk. Continue to grow the amount of this content each year and you are faced with a problem that cannot be ignored.

IBM FileNet P8 Platform offers enterprise-level scalability and flexibility to handle the most demanding content challenges, the most complex business processes, and integration with many of an organization's existing systems.

1.1.1 The problem defined

Structured data refers to tabular information that can be stored in a Relational Database Management System (RDBMS). Examples of structured data include a customer record or a transaction history with well-defined data fields, such as customer name, customer ID, and transaction date. Search facilities within an RDBMS can quickly locate records based on an index of the data fields.

Unstructured data, on the other hand, does not lend itself to fitting in an orderly fashion within the columns and rows of a database. Unstructured data usually resides within documents, electronic forms, reports, Web pages, and the bodies and attachments of emails. Unstructured data can be found spread across file shares, intranets, e-mail systems and users' desktops and consists of customer correspondence, contracts, newsletters, press releases, loan or job applications, process documentation, and myriad of other forms of communication. Access to this type of data is typically provided through file and Web browsers and through the client applications in which the files were authored. Unstructured data now makes up the overwhelming majority of the new content being created.

The difficulty in accessing and managing unstructured data is that it inherently lacks the normalization and standardization of its structured counterpart, along with the controls to lock down access, easily index and search, and to ensure that data is current. With information embedded inside documents and a wide

range of other file types, unstructured data poses a number of *challenges* to organizations that are trying to manage it as a strategic asset, which includes:

- ▶ **Scope:** Unstructured data is strewn across the enterprise, with no easy means of querying it all. With so much unstructured content out there, knowledge workers are easily inundated with meaningless search results that dilute the value of important information and prevent them from finding what they actually need.
- ▶ **Searching:** As a minimum requirement, unstructured content must be full-text indexed to enable the location of information buried in documents and files, something that has traditionally required different applications and interfaces for each content type. Many applications require richer content analytics, such as extracting and indexing formal names of people, businesses, places, and other entities.
- ▶ **Storage:** While there are a host of regulations that mandate the retention of content, many organizations found it easiest to simply hang on to everything. As a result, storage costs continue to grow, driven in large part by keeping out-of-date, irrelevant, and often redundant documents and files.
- ▶ **Context:** Many critical business processes are heavily driven by unstructured content, yet maximizing the value of this information within the context of the processes being performed is difficult, if not impossible. In addition, obtaining a complete view that considers all information sources, both structured and unstructured, on which to base decisions is a major challenge.
- ▶ **Risk:** Any organization might be subject to litigation. Organizations must be prepared to produce electronic evidence of business activity in response to an audit or litigation. This process might involve the identification and production of vast amount of documents, e-mails, and other content, regardless of where they are housed, which is an overwhelming task without centralized management. Furthermore, companies that retain obsolete and unimportant content are not only driving up legal discovery costs through the unnecessary review of extraneous data, but are also exposing themselves to the risk of having this potentially damaging information considered as evidence against them.
- ▶ **Security:** Restricting access to confidential, proprietary, and other sensitive content is greatly complicated when files are spread across many systems and storage environments that lack essential management facilities.
- ▶ **Accuracy:** File shares represent a widely-used approach to centralizing access to documents and other information. Unfortunately, file shares offer no effective means of managing iterative versions of content, which makes it possible for outdated information to exist and for newer revisions to be overwritten. File shares also do not support designating and controlling approved versions for document life cycle support. There is a lack of process controls to ensure that only approved content is made available for

consumption by an organization's knowledge workers and updated when needed.

- ▶ **Efficiency:** When employees do not have a complete picture of the intellectual property that exists within an organization, valuable knowledge cannot be leveraged and a high degree of re-work can exist as employees continually re-invent the wheel. Staff can also waste a great deal of time locating information when it is not properly managed and made accessible.
- ▶ **Availability:** Like users, business applications often require access to data that resides in multiple, distinct repositories that were never intended to share their content with other systems. One common approach to try to address these silos is to simply duplicate electronic content for each system that needs to consume it, which introduces yet another set of information efficiency, storage, and maintenance issues.

Summary: Today's challenges with exponential growth of unstructured and semi-structured data are managing the *scope* of the data, *searching* of data, *storage* of the content, getting values from the *context* of the data, potential *risk* associated with the data, and *security* requirement of the data, ensuring *accuracy* of the data, *efficiency* in retrieving the data, and making data *available* to those who need it.

1.2 Enterprise content management and its capabilities

To address the challenges of managing unstructured and semi-structured content, enterprise content management (ECM) is introduced as the management of these types of content at the enterprise level. The management includes capturing, organizing, securing, and storing unstructured and semi-structured content within an enterprise from creation to disposition, and facilitates searching and controlled access to managed information assets directly from business processes or from other line-of-business applications.

ECM is not a product or a specific technical solution. Rather, it is a disciplined framework for identifying the sources and consumers of an organization's information, applying controls to its creation, use, and handling, and making it available to those applications and workers who need it to complete the business processes and functions for which they are responsible.

The objective of ECM is to ensure that access to the information that is being shared across an organization is timely, accurate, and secure, and provides the required processes to execute key functions in support of strategic business goals. ECM is about empowering employees at all levels of an organization to

make the right decision at the right time, and applies with equal relevance to Government, insurance, manufacturing, retail or any other industry.

ECM: ECM is about empowering people to make decisions better and faster.

1.2.1 Enterprise content management capabilities

ECM represents an evolution and convergence of a variety of different technologies that include elements of document management, imaging, workflow, collaboration, knowledge management, and compliance. From a functional perspective, there are a number of fundamental capabilities that are necessary to support effective managing and sharing of content:

- ▶ Support for common file formats
- ▶ Flexible authoring capabilities using common desktop authoring tools
- ▶ Ability to allow or restrict specific access and action permissions
- ▶ Fulltext indexing of unstructured content and the ability to search on the content and its metadata
- ▶ Versioning of content with check in and check out feature
- ▶ Workflow process integration
- ▶ Ability to access content through Web and portal integration

ECM goes beyond these core features to meet a broader range of requirements, adding:

- ▶ Capture and collect both physical and electronic content
- ▶ Support for virtually any file format
- ▶ Integrate with full business process management
- ▶ Support for an enterprise taxonomy
- ▶ System-wide audit and tracking capability
- ▶ Content transformation
- ▶ Content life cycle management, from creation to archival or destruction
- ▶ Federated management and collection of content across repositories
- ▶ Open interfaces to integrate with other applications and systems and deliver highly-specialized applications
- ▶ Integrated security and access management

- ▶ Robust metadata support
- ▶ Automated classification of existing and new content
- ▶ Enterprise-level availability and scalability

In addition to the key elements that we previously listed, there is a logical move toward supporting an organization's records management and eDiscovery requirements as an extension of an enterprise content management system. This move entails expanding on traditional content life cycle management capabilities to identify documents, e-mails, and other files that must be declared as records to enable defined retention periods for specific types of content, the ability to place records on hold to temporarily stop their disposition, and automating legal discovery activities across the entire enterprise.

1.3 IBM FileNet P8 Platform

A *platform* serves as a foundation on which applications can be built and deployed. A software platform provides interoperability to a wide selection of database, operating system, storage, security, and Web server environments, thereby reducing integration costs and improving efficiency. Using the platform approach can effectively address the complex demands of managing content across an enterprise.

Implementing a platform for a company's enterprise content management provides the following benefits:

- ▶ A unified, open architecture for SOA environments, simplifying application deployment and lowering the total cost of ownership for the geographically distributed enterprise.
- ▶ A means to cost effectively deploy and manage IT services in a dynamic business environment while adhering to standards for security, storage, and service delivery.
- ▶ A centrally controlled and managed infrastructure that allows ECM capabilities to be provisioned in a distributed manner to meet content and process-driven business needs.
- ▶ Reusable systems capabilities that enable content and related business processes to be used in multiple business applications.

IBM FileNet P8 Platform is the unified enterprise foundation for the integrated IBM FileNet P8 products. It provides the core components that the add-on IBM FileNet P8 products seamlessly interoperate, sharing a common information infrastructure and associated security model, taxonomy, and set of Application Programming Interfaces (APIs). IBM FileNet P8 applications leverage the Java™

Enterprise Edition application model to build multi-tier applications that deliver the scalability, accessibility, and administration that enterprise applications require.

The core of IBM FileNet P8 Platform is provided by the following three products:

- ▶ IBM FileNet Content Manager
- ▶ IBM FileNet Business Process Manager
- ▶ IBM FileNet Records Manager

The main components that built the core products are the following engines:

- ▶ Content Engine
- ▶ Process Engine
- ▶ Application Engine

Each engine consists of a collection of services and applications that perform well-defined sets of services and tasks. We discuss these engines in 1.3.4, “IBM FileNet P8 architecture and its core engines” on page 17, and explore them in greater depth in Chapter 2, “Core component architecture” on page 27.

IBM FileNet P8 Platform: IBM FileNet P8 Platform is a unified enterprise content management and business process management platform that removes the need to support separate solutions for content, process, and compliance applications. This tight integration increases operational efficiency by reducing the number of products and vendors across the enterprise, while providing a single, robust interface for building and deploying content and process applications.

Specific benefits that are available to organizations that adopt IBM FileNet P8 as their ECM platform are:

- ▶ Combines an enterprise content management reference architecture and core enterprise platform with comprehensive business process management and compliance capabilities.
- ▶ Includes a comprehensive set of content and process management business services that can be consumed and deployed in a service-oriented architecture.
- ▶ Supports a flexible API for Java, Microsoft® .NET, and XML Web services application development for a rich and interactive user experience that is easily customized.
- ▶ Delivers data center manageability and support for enterprise system management tools and enterprise scalability and flexible system deployment in clustered and highly-available environments.

- Provides distributed content caching and network optimization features that provide optimal performance across geographically distributed sites and wide area networks.
- Includes multilingual system capabilities for decentralized, federated system architectures, advanced security services, comprehensive auditing, and a standards-based authentication framework.
- Integrates content and process directly into an organization's primary desktop applications and most important enterprise business functions.

1.3.1 IBM FileNet P8 solutions by industry

IBM FileNet P8 is successfully utilized to address real business problems for thousands of customers in various industries. Table 1-1 lists the IBM FileNet P8 solutions by industry.

Table 1-1 IBM FileNet P8 solutions by industry

Financial	Insurance	Government	Manufacturing	Cross industry
Account Origination	Claims	Health & Welfare	Life cycle Asset	
Management	Customer Service			
Loan Processing	Life and Annuities	Court Case Management	Compliance	Knowledge Management
Cash Management	Property and Casualty	Human Services	Quality and Safety	Procurement
Mortgage Loan Processing	Underwriting	Unemployment Case Management	Product Documentation Management	HR Onboarding

Expanding on this enterprise-ready platform is a wide range of specific solutions that are developed to take advantage of the core strengths and capabilities of the IBM FileNet P8 Platform, which includes both standard and custom business applications. Combining the inherent content and process facilities of IBM FileNet P8 into many high-value solutions, IBM enables organizations to introduce an enterprise architecture that they can extend to accommodate a host of critical business requirements, which increases the value of the platform with the addition of each new application.

1.3.2 Core IBM FileNet P8 products

The main focus of IBM FileNet P8 Platform is content management, business process management, and compliance. The following products address these areas:

Core IBM FileNet P8 products:

- ▶ IBM FileNet Content Manager
- ▶ IBM FileNet Business Process Manager
- ▶ IBM FileNet Records Manager

Together, they form the core systems that support all of the other applications that are built on the platform.

IBM FileNet Content Manager

IBM FileNet Content Manager is the core enterprise content management product for the IBM FileNet P8 Platform. It serves as the main content management, security management, and storage management engine for the family of IBM FileNet P8 products. IBM FileNet Content Manager combines universal content management and advanced document management capabilities with industry-leading *active content* process capabilities to deliver active management of all types of content across the enterprise, regardless of the format or the repository in which it resides. IBM FileNet Content Manager maintains secure control over metadata and process and compliance activities, while managing highly customized, transactional content and supporting extensive versioning and parent-child capabilities, approval workflows, and integrated publishing support. Also provided are the tools to deploy turnkey solutions that handle complex and compound document types using out-of-the-box workflow and process capabilities. You can install and implement IBM FileNet Content Manager as a stand-alone product.

For more information, refer to the product manuals and the following IBM Redbooks publication:

- ▶ *IBM FileNet Content Manager Implementation Best Practices and Recommendations*, SG24-7548

IBM FileNet Business Process Manager

IBM FileNet Business Process Manager uses the native process engine within IBM FileNet P8 to manage workflow among people and systems for content and human-centric processes and to integrate directly with external applications. IBM FileNet Business Process Manager is proven in organizations of all sizes to increase process performance and consistency, reduce cycle times, and improve productivity and decision making by automating, streamlining, and optimizing

processes. In addition to facilitating extensive automation, IBM FileNet Business Process Manager also provides supporting tools for comprehensive process management, which includes process modeling, advanced analytic and simulation, and the business activity monitoring for real-time process performance awareness. IBM FileNet Business Process Manager supports process standards, such as Business Process Modeling Notation (BPMN) for modeling and the XML Process Definition Language (XPDL) for definition and execution. Leveraging another important aspect of the IBM FileNet P8 portfolio, IBM FileNet Business Process Manager can use electronic forms as a means of capturing critical process data and serving as a task interface. You can install and implement IBM FileNet Business Process Manager as a stand-alone product.

For more information, refer to the product manuals and the following IBM Redbooks publication:

- *Introducing IBM FileNet Business Process Manager*, SG24-7509

IBM FileNet Records Manager

IBM FileNet Records Manager streamlines records-based activities to enforce compliance either with or without user participation. You can use IBM FileNet Records Manager to classify, apply holds and retention policies, and to store electronic records according to fiscal, legal, and regulatory requirements. IBM FileNet Records Manager, in conjunction with a powerful family of supporting tools, securely manages the declaration, classification, security and access, auditing and monitoring, authenticity, preservation, and disposal of electronic and physical records. All of this is made possible by exploiting the underlying business process management capabilities of the IBM FileNet P8 Platform to execute defined business rules to classify data and apply retention rules automatically to important corporate documents. Using a concept known as *federation*, which we discuss in more detail in 1.3.6, “Content federation” on page 19, IBM FileNet Records Manager can apply record controls to documents and content that do not reside in an IBM FileNet P8 content repository.

For more information, refer to the product manuals and the following IBM Redbooks publication:

- *Understanding IBM FileNet Records*, SG24-7667

1.3.3 IBM FileNet P8-driven expansion products and applications

Although the list of IBM FileNet P8-driven products and applications continues to grow, the following list provides a short description of a few that are currently available that help organizations that standardized on IBM FileNet P8 to

automate processes, improve productivity, and drive out cost by building on their investment in an enterprise content platform:

- ▶ Content ingestion related products:
 - IBM FileNet Content Collector
 - IBM Capture Professional and ADR
- ▶ Connectors and federation products:
 - IBM FileNet Application Connector for SAP R/3 (ACSAP R/3)
 - IBM Content Integrator
 - IBM FileNet Services for Lotus Quickr
 - IBM FileNet Connectors for Microsoft SharePoint
- ▶ Application framework products:
 - IBM FileNet eForms
 - IBM FileNet Business Process Framework
 - IBM FileNet Business Activity Monitor (BAM) and Cognos Now
- ▶ Search, classification, and discovery products:
 - IBM Classification Module
 - IBM Content Analyzer
 - IBM eDiscovery Manager and Analyzer
 - IBM OmniFind Enterprise Edition
- ▶ Other expansion product:
 - IBM FileNet System Monitor

There are many more applications that support the IBM FileNet P8 Platform that are beyond the scope and constraints of this publication. We provide a link to information about these applications and partner products at the end of this section.

IBM FileNet Content Collector

For situations where federation might not be the most effective approach for managing diverse content sources, IBM Content Collector provides powerful ingestion tools for collecting, enhancing, and managing all types of content, regardless of storage location. IBM Content Collector is typically a preferred approach for content sources that are not scalable, lack security and proper controls, or do not support lockdown, allowing loosely managed content to be moved from their existing location (file system or email servers) into IBM FileNet P8 repository. IBM Content Collector is an ideal solution for reining in file shares, e-mail stores, and other systems to ensure proper management of business content that is identified as being of importance to the organization.

It comes with two offerings, IBM Content Collector for File Systems and IBM Content Collector for Emails:

- ▶ **IBM Content Collector for File Systems**

Used to collect files in file systems. It replaces IBM FileNet Records Crawler.

- ▶ **IBM Content Collector for Emails**

Used to collect emails from e-mail databases and non-e-mail documents in e-mail databases. It replaces IBM FileNet Email Manager and IBM Content Manager CommonStore products. Rather than simply saving all messages, IBM Content Collector for Emails provides intelligent, rule-based retention policies that deliver enforced archive management while ensuring that regulatory compliance is maintained. It addresses the operational problems that are introduced by the growing size of e-mail and electronic messaging data stores with records management, legal discovery, supervision, and monitoring for non-compliance. IBM Content Collector for Emails assists in managing mailboxes, increases server performance, enables faster backup and restore, provides for easier upgrades, and leverages storage management best practices for e-mail. IBM Content Collector for Emails supports Lotus® Domino®, Microsoft Exchange, and Novell® GroupWise.

IBM Capture Professional and ADR

IBM Capture Professional and Capture Advanced Document Recognition provide paths to ingest paper documents, faxes, and other content images. Corporations use these products to replace manual mailroom activities and expand that functionality with automated classification and other document handling activities. In conjunction with the IBM FileNet P8 Platform, paper documents can become active content by starting or joining work processes.

IBM FileNet Application Connector for SAP R/3 (ACSAP R/3)

IBM FileNet Application Connector for SAP® R/3 (ACSAP R/3) is a modular ECM solution that integrates tightly with mySAP™ applications through the SAP ArchiveLink™ and Kpro interfaces. It provides outbound archiving and retrieval for SAP-generated business documents, SAP database data and reports, and inbound archiving and retrieval for documents that are generated outside of SAP. It supports early, late, and simultaneous document linking scenarios, which includes barcode processing. In addition to satisfying typical SAP R/3® document-enabling use cases, document linking to IBM FileNet P8 repositories enables you to use IBM FileNet ECM solutions to improve and extend the business processes beyond the confines of SAP applications.

IBM Content Integrator

IBM Content Integration (previously known as IBM Information Integrator Content Edition) incorporates content in existing repositories and makes that

information available to the IBM FileNet P8 Platform system. Using Content Integrator, businesses reuse existing records, which enables them to keep their current investments while expanding their usefulness. Documents can become part of active processes, indexed, classified, and all of the other features that the IBM FileNet P8 Platform and expansion products offer.

IBM FileNet Services for Lotus Quickr

IBM FileNet Services for Lotus Quickr™ join the teamwork and collaboration of Lotus Quickr with the active content and business process management of IBM FileNet. Lotus Quickr provides Web user interface, team places, and integration with the desktop, e-mail, document creation, and other applications. You can act as a team to accomplish your tasks in your native environments, whether this is using Lotus Notes® or Microsoft Outlook®, Microsoft Word, Sametime®, or other products. This powerful combination accelerates productivity while safeguarding critical corporate assets.

IBM FileNet Connectors for Microsoft SharePoint

IBM FileNet Connectors for SharePoint® integrates the power of the IBM FileNet P8 Platform with the team collaboration capabilities of Microsoft SharePoint. This integration enables you to stay in the familiar Microsoft Office environment and leverages IBM FileNet P8's more comprehensive ECM capabilities. It comes with two offerings, IBM FileNet Connector for Document Libraries and IBM FileNet Connector for SharePoint Web Parts:

- **IBM FileNet Connector for Document Libraries**

This connector sweeps content into IBM FileNet Content Manager based on administrator-configured metadata and then moves the Microsoft SharePoint content to the IBM FileNet P8 repository. Content that is originally created in Microsoft SharePoint, and now managed by IBM FileNet P8, is accessible to authorized users and can be placed under full life cycle and compliance management. The solution helps to enforce appropriate content, process, and compliance management disciplines on Windows® SharePoint Services (WSS) content, while also enabling SharePoint Portal users to search non-SharePoint content, providing one place to go for their information retrieval needs.

- **IBM FileNet Connector for SharePoint Web Parts**

This connector extends Microsoft SharePoint to allow document management and process management activities to be completed against content that resides in IBM FileNet P8 within the Microsoft Office SharePoint Server (MOSS) environment. You can author, browse, and manage content while having seamless access to your IBM FileNet P8 workflow inbox for task management activities.

IBM FileNet eForms

IBM FileNet Electronic Forms (eForms) are, in practice, the basic requirement for modern businesses. Customers expect to be able to interact with corporations using the Internet and other software interfaces. These relations require sophisticated, easy to use forms that gather the required information without repetition and frustration. Forms need to be smart: using database lookups, automated fill ins, and cascading interactions to provide personalized responses. IBM FileNet eForms provides an easy to use designer and multifaceted and comprehensive functionality to meet ever changing market requirements.

IBM FileNet Business Process Framework

IBM FileNet Business Process Framework (BPF) accelerates application development on business process management with a structure of user interface, workflows, and inboxes that focus on case management. Case management, such as mortgage application and support, is a frequently implemented application in various forms on the IBM FileNet P8 Platform, and using this structure saves months of time and hundreds of thousands of dollars. By providing well-designed and tested solutions, clients focus on configuration rather than coding.

IBM FileNet Business Activity Monitor (BAM) and Cognos Now

IBM FileNet Business Activity Monitor (BAM) is a flavor of Cognos® Now that was originally out sourced to Cognos. They provide visibility into the business process and performance by showing real time measurements of the system. Now that Cognos is part of the IBM family, they are sometimes spoken of interchangeably, but the key difference is that the IBM FileNet BAM comes with pre-configured variables so that monitoring of the system is ready out-of-the box, which saves set up time.

IBM Classification Module

IBM Classification Module combines text analysis classification with rules-based approaches to automate the organization of unstructured content by analyzing the full text contents of documents and e-mails. IBM Classification Module allows more content to be brought under management, categorized more accurately, without burdening authors or subject matter experts with extra categorization tasks, while mitigating compliance risk by ensuring that records management practices are uniformly followed.

For more information about IBM Classification Module, refer to the product manuals and the following IBM Redbooks publication:

- *IBM Classification Module, SG24-7707*

IBM Content Analyzer

IBM Content Analyzer (formerly known as IBM OmniFind® Analytics Edition) discovers new insights from unstructured and structured content to enable better business decisions. IBM Content Analyzer integrates with existing IBM FileNet P8 content repositories and Business Intelligence (BI) tools to augment existing BI environments with analytic and reporting on ECM-managed content.

IBM Content Analyzer powers additional insights for solutions in customer care, new product innovation, and early problem detection using the open Unstructured Information Management Architecture (UIMA) standard.

For more information about IBM Content Analyzer, refer to the product manuals and the following IBM Redbooks publication:

- *Introducing OmniFind Analytics Edition: Customizing Text Analytics*, SG24-7568

IBM eDiscovery Manager and Analyzer

IBM eDiscovery Manager is a highly-scalable solution for e-mail that manages electronic discovery and optimizes litigation response. eDiscovery Manager enables authorized IT and legal users with targeted eDiscovery functions to search, cull, hold, and export case-relevant email. eDiscovery Manager supports secure e-mail evidence preservation in an audit-tracked repository of record with chain of custody and integrates directly with the IBM FileNet Records Manager offering. eDiscovery Manager employs IBM e-mail archiving solutions for Lotus Domino and Microsoft Exchange Server and exploits IBM Enterprise Content Management repositories as a litigation vault, exporting case-relevant e-mails for further detailed review.

IBM eDiscovery Analyzer provides conceptual search and analysis of cases that IBM eDiscovery Manager creates. eDiscovery Analyzer enables legal professionals and support specialists to quickly refine, analyze, and prioritize case-related e-mails, providing insight into a case and helping to dramatically reduce eDiscovery review costs. eDiscovery Analyzer provides full security and auditability of case materials, including a privilege model for case access and an audit trail of all actions.

IBM OmniFind Enterprise Edition

IBM OmniFind Enterprise Edition is a search solution that focuses on highly scalable, available, and secure search. This solution provides access to content for millions of documents and thousands of users with pre-built integrations for indexing data and content. It also provides a platform for constructing semantic search and content analytics solutions, such as entity analytics, sentiment analysis, threat analysis, global name recognition, and other search

requirements. IBM OmniFind Enterprise Edition makes content more usable by making it searchable.

IBM FileNet System Monitor

IBM FileNet System Monitor monitors the health of the entire IBM FileNet system environment to increase uptime and to help IT meet service level agreements (SLA). IBM FileNet System Monitor integrates with industry-leading enterprise systems management tools, which includes IBM Tivoli®, HP OpenView, BMC Patrol, CA Unicenter, and Microsoft, allowing data center and help desk staff to remotely track the health of IBM FileNet and dependent servers. IBM FileNet System Monitor proactively monitors more than 1,100 performance and system parameters across the IBM FileNet product family, automating more than 75 manual IBM FileNet system administrative tasks to increase productivity and reduce IT support costs. Pre-defined corrective actions significantly impact troubleshooting efforts, and administrators can receive alerts using a variety of notification methods, such as Blackberry, pager, cell phone, e-mail, or Web console. System Monitor also maintains a knowledge base of historical information for analysis and reporting.

Additional references

To read more about these and other IBM ECM products and solutions, go to <http://www.ibm.com/software/data/content-management>

To read available IBM Redbooks publications on ECM-related products and solutions, go to the following Web site and type **ecmredbooks** in the New Search field:

<http://www.redbooks.ibm.com>

In addition to the list of applications that we previously mentioned, one of the biggest testimonials to the benefits of the flexible IBM FileNet P8 architecture is the large number of partner solutions that are built using IBM FileNet P8 as the foundation. Take time to look at the Worldwide ECM Partner Solutions Handbook:

ftp://ftp.software.ibm.com/software/data/ECM/Bro/IBM_ECM_WW_Partner_Solution_Handbook_v3.pdf

The Worldwide ECM Partner Solutions Handbook has more than 200 IBM Enterprise Content Management solutions from business partners covering Compliance, Health Information Management, Credit Risk Management, Legal Case Management, Claims Processing, Invoice Processing, Pension Administration, and many more areas. Each of these is built on the IBM FileNet P8 Platform and offers a way to introduce complete ECM solutions without the time and risk of developing a custom application from scratch.

1.3.4 IBM FileNet P8 architecture and its core engines

The IBM FileNet P8 products share an underlying architecture that is comprised of a set of services or engines, which enables a IBM FileNet P8 solution to lower total cost of ownership by requiring fewer servers and minimizing integration efforts.

Working in concert at the core of the IBM FileNet P8 architecture are the Content Engine, Process Engine, and Application Engine, as shown in Figure 1-1.

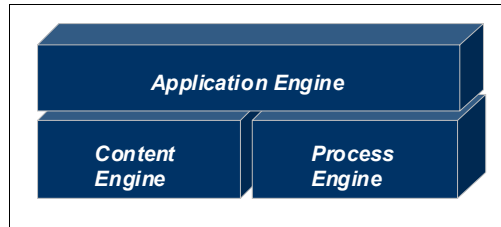


Figure 1-1 Core IBM FileNet P8 engines

While each of the engines are explained in further detail in Chapter 2, an overview of each is provided here to convey the importance of a modular architecture with distinct services that are dedicated to addressing the primary requirements of enterprise content management.

The Content Engine

The Content Engine provides main library services, manages documents, folders, content, and business-specific objects, and allows content to be stored, retrieved, transformed, classified, and secured. The Content Engine can manage content stored in a file store, a database, or a fixed storage device.

Content Engine Service manages interactions with the IBM FileNet P8 Content Search Engine, allowing text-based searches to be performed against the contents of documents and their properties (metadata).

Content Engine's object-oriented, extensible metadata model enables complex and flexible data representation, allowing a high degree of content and application reuse. Java, .Net, and a Web Services application programming interface provide an extensible platform for development of cross-repository solutions.

Using the Content Engine, you can file documents into folders for logical separation above and beyond what the metadata provides alone. Filing documents in multiple folders does not create extra copies of those documents; instead, it creates a logical association between the folder and the document.

The Process Engine

The Process Engine incorporates software services for managing all aspects of business processes (also called workflows), such as process execution, process routing, rules management, process simulation and modeling, and workflow analysis. The key components of the Process Engine are:

- ▶ **Process service:** The core of the business process management system. It provides workflow services on behalf of the Process Engine.
- ▶ **E-mail notification:** Enables automatic transmission of e-mail to users when specified process-related events occur. E-mail notification can also be used to track workflows.
- ▶ **Rules connectivity framework:** Provides a framework for rules integration. A process designer or business analyst creates business rules (a third-party application purchased separately from the Process Engine) and associates the rules with the steps of a workflow.
- ▶ **Process Task Manager:** A tool provided by the Process Engine that provides administrative tools for configuring and managing process-related services on the Process Engine server.
- ▶ **Process Analyzer:** A tool provided by the Process Engine that provides analysis capabilities to determine cycle times, find bottlenecks, and generate reports and charts to analyze the processes deployed in the Process Engine system.
- ▶ **Process Simulator:** A tool provided by the Process Engine that allows business analysts and process owners to simulate processes or perform what-if scenarios with hypothetical or historical data.
- ▶ **Component Integrator:** Provides an extensible integration framework that allows additional connectors to be created for the execution of external Web services or code. Java and JMS adaptors are provided out-of-the-box.
- ▶ **Process Web Services:** Provides a Web services API to the Process Engine.

Defined processes are stored in a Content Engine repository, which allows the life cycle of a process definition to be managed by controlling access and tracking different versions of the same process.

The Application Engine

The Application Engine provides the presentation layer and includes out-of-the-box user interfaces and components for building custom solutions. The Application Engine is the component that hosts the WorkplaceXT Web application, Workplace Java applets, and application development tools.

WorkplaceXT is a browser-based client application that is written with the IBM FileNet APIs that are included with the Content Engine. Workplace provides an

interface for adding content to the IBM FileNet P8 system and for performing other primary content-oriented tasks, such as declaring records, accessing workflow queues, and searching. Workplace is built using the IBM FileNet Web Application Toolkit and runs within a Web Container on a J2EE application server.

The Application Engine also supports IBM FileNet P8's integration with Microsoft Office, WebDAV, the Content and Process APIs, and the Web Application Toolkit.

1.3.5 Support for Java, .NET, and XML Web Services Frameworks

IBM FileNet P8 includes an extensive collection of development tools that span the content and process management capabilities that we outline in this document:

- ▶ Graphical tools for defining and designing application components, such as processes, metadata definitions, searches, and templates.
- ▶ Java APIs for programmatic access to content and process capabilities.
- ▶ A .NET API for developing Content Engine applications.
- ▶ SOAP-based Web services for building solutions using a service-oriented architecture (SOA) that can execute on a wide range of platforms and can use a variety of languages and toolkits to access most of the functionality that is available through the Content Engine and Process Engine Java APIs.
- ▶ Integrations with leading portal vendors for building Web-based applications.
- ▶ User interface elements that can be reused in custom applications.
- ▶ Code module capabilities where Java classes containing event action code are stored in the repository and are therefore easily deployable.

1.3.6 Content federation

More and more, companies are recognizing the need to connect multiple point solutions for content management to gain a more comprehensive picture of enterprise content. An *integration* strategy offers the quickest path to a unified view of unstructured content by providing a single point of access to content that is scattered across multiple repositories. Integration is a good strategy for high-volume environments, allows customers to leave content in place, and provides a repository-independent approach to federation. A *standardization* strategy is an excellent day-forward strategy for customers who want to select IBM as their standard ECM platform. This approach allows customers to leverage the IBM FileNet P8 Platform to manage new and historical content from multiple repositories. A *consolidation* strategy is best suited to customers for

whom the advantages of a single ECM repository outweigh the costs and risks of migrating content off of their heritage repositories.

A toolset called IBM Content Integrator (formerly known as IBM WebSphere Information Integrator Content Edition) provides connectivity to content that is stored in third-party repositories. Content Federation Services (CFS) Server for Content Integrator stores metadata from these repositories in the IBM FileNet P8 Content Engine, which enables true content management federation.

Content Integrator's integration services provide a single, consistent interface to the underlying content repositories, including content functionality and workflow capabilities. These integration services expose a super-set of content management and workflow functionality and also maintain the awareness of both the available repositories and the functional capabilities of each repository.

Figure 1-2 illustrates the Content Integration services for IBM FileNet P8.

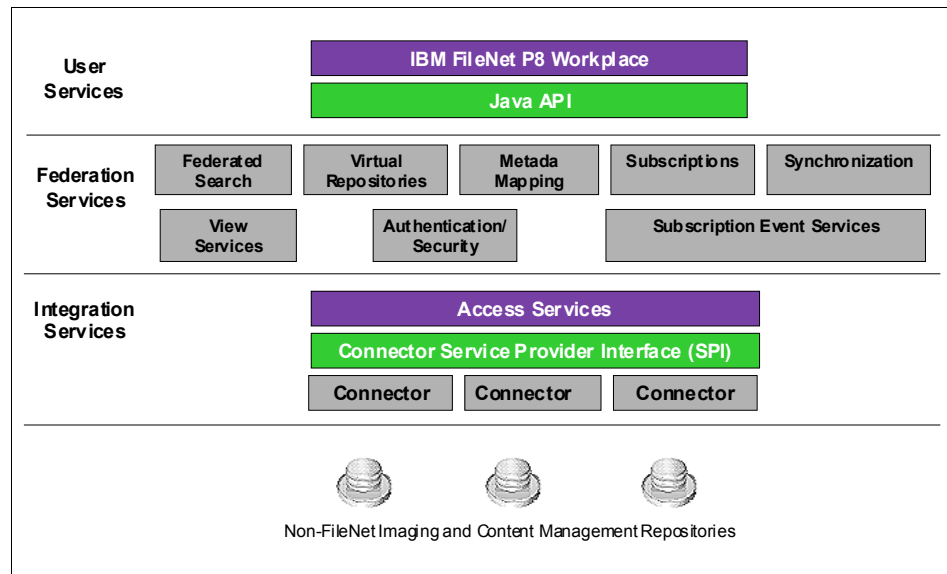


Figure 1-2 Content Integration services for IBM FileNet P8

As foundational components of the IBM content federation strategy, CFS, and Content Integrator enable out-of-the-box capabilities for integrating content from one or more third-party repositories into IBM FileNet P8:

- ▶ Search for content: Perform parametric and full-text searches against one or multiple content repositories.
- ▶ Capture content: Add content and metadata to repositories.

- ▶ Control content: Perform library functions, such as check-in/check-out and copy or transfer folders and documents within a repository or across repositories while maintaining properties, versioning information, and other content attributes.
- ▶ Retrieve content: Retrieve content and associated metadata values from repositories in the content's native format or in an XML document.
- ▶ Update content: Make changes to content and update metadata values, annotations, and security settings while maintaining version control.
- ▶ Manage content hierarchies: Create and delete folders, file and un-file content in folders, retrieve folder contents, and update folder properties.
- ▶ Search for work items: Perform parametric searches against one workflow engine or federated searches against multiple workflow engines.
- ▶ Create new work items: Initiate new instances of workflow processes and apply metadata values and content attachments.
- ▶ Retrieve work items: Retrieve work items and any attached content from an in-box or specific queues or steps in the workflow process.
- ▶ Update work items: Make changes to work items including metadata and attachments. Perform actions on the work item, such as locks, suspend, resume, and dispatching.
- ▶ Audit: All actions initiated through Content Integrator can be audited at various levels with all of the pertinent information, such as the time, the user, the specific action taken, and the item being accessed.
- ▶ Maintain security: Ensure that users access only authorized content and work items by taking advantage of the security features that are inherent in the underlying system.
- ▶ Manage sessions: Log on and log off to content repositories and workflow systems with password encryption over the wire.

In combination with the IBM FileNet P8 Catalog Management Services, Content Federation Services enable IBM FileNet P8 customers to access content from select heterogeneous repositories and to truly federate this information to provide a single enterprise source for critical business content. This content can then be exposed to any external business application through the IBM native XML Web Services. Similarly, this content and its associated metadata can also be accessed and consumed by the various product suites that are available for the IBM FileNet P8 Platform, for example, this federation enables the IBM FileNet Records Manager application to manage records content that remains in external repositories, which includes other document and content management systems. By managing content in place, IBM FileNet P8 allows compliance practices to be centralized, providing visibility and control over all record sources.

In addition, this federation approach protects existing investments in other content stores.

IBM FileNet Content Federation Services for Image Services (CFS-IS) integrates and federates content from the Content Engine and IBM FileNet Image Services repositories. CFS-IS enables the Content Engine to use Image Services as another content storage device. Users of IBM FileNet P8 applications have full access to content that is stored in existing Image Services repositories. Anything that is created in WorkplaceXT or created programmatically using the Content Engine APIs can be stored in the Image Services permanent storage infrastructure. Existing Image Services content is preserved and usable by Image Services heritage applications but reusable by IBM FileNet P8 applications, such as Workplace and Records Manager, without duplication and without change to existing applications. The location of document content is transparent to all applications.

1.3.7 Performance and scalability

IBM FileNet P8 has been performance tested against an object store containing in excess of 2 billion objects. Exercising standard Content Engine functionality including browsing, document creation, check in and check out, search and retrieval, along with the primary Process Engine tasks of workflow launches and various queue functions, the tests showed overall average atomic transaction response times of well under a second on modest hardware. Server utilization remained well within acceptable tolerances and never exceeded 26 percent on any individual IBM FileNet P8 engine.

Further proof of IBM FileNet P8's ability to handle large loads comes from the IBM customers. Actual production implementations of IBM FileNet P8 include companies with more than 50,000 users and repositories with hundreds of millions of documents, with tens of millions more being added monthly.

The IBM FileNet P8 components support enterprise-level scalability with a multi-tier, distributed architecture, supporting production environments with tens of thousands of users. To accommodate various scalability needs, IBM FileNet P8 offers both vertical and horizontal scalability solutions. The vertical scalability of a server can be defined as its ability to handle additional workload by the addition of a proportional amount of processing power. Horizontally scalable systems can handle additional workload by increasing the size of the server farm as the workload increases.

Among the options for scaling an IBM FileNet P8 system are:

- ▶ Application Engine:
 - Farmed (scaled horizontally) to support increased workload requirements.
 - Scaled vertically by either running multiple instances of a single Application Engine version on a single box or by configuring a single instance to leverage system resources.
- ▶ Content Engine:
 - Farmed (scaled horizontally) to support increased workload requirements.
 - Scaled vertically by either running multiple instances on a single box or by configuring a single instance to leverage system resources.
- ▶ Process Engine:
 - Farmed (scaled horizontally) to support increased workload requirements.
- ▶ Database software can be remotely configured to run on a separate machine.

1.4 Enterprise Reference Architecture

Enterprise Architectures (EAs) provide a service-oriented framework for visualizing how the various elements of an IT infrastructure align with the goals of the business. Using an EA, organizations can map their plans for integrating key back end systems with line-of-business applications to meet strategic goals. Such a map serves as a business tool to allow organizations to identify disconnects between IT and corporate objectives and to plan for change. Enterprise Architectures consider business, data, technical, and application architectures.

A key benefit of an EA is the visibility that it provides into all systems, offering both technical and business planners a complete view of an organization's requirements and capabilities. This visibility enables Enterprise Architects to define architectural solutions, frameworks, patterns, and reference architectures for use across multiple systems within the organization. Ultimately, an EA saves money by helping to promote consistency across systems and guiding development teams toward using a common set of proven approaches to application architecture.

An *Enterprise Reference Architecture* (ERA) is a blueprint for designing information technology. It defines a set of building blocks and shows how they fit together, providing a common vocabulary and conceptual framework for information technology environments. IT organizations are increasingly using Enterprise Reference Architectures as mechanisms to define the key services

that their IT environments must deliver and for providing guidelines for IT planning and technology selection and implementation.

IBM developed a unified ERA for IBM FileNet P8 enterprise content management (ECM) and business process management (BPM). The purpose of this reference architecture is to:

- ▶ Facilitate alignment of IT and business priorities
- ▶ Highlight the integral relationship of ECM, BPM, and compliance
- ▶ Show how the completeness of the IBM FileNet P8 technology vision can help build out an organization's own ERA
- ▶ Conceptualize a solution involving IBM FileNet P8 and identify common integration points with common enterprise technology

The IBM FileNet P8 ERA provides a comprehensive view of an organization's strategic infrastructure around ECM and BPM, simultaneously considering both data and process needs. The IBM FileNet P8 ERA affords visibility and direction into the technologies and services that can be utilized to guide future application design and development. The development of this joint business- and technical-level vision ensures that the critical content and process capabilities of the platform are used to their fullest advantage within a system-oriented architecture.

The architecture framework includes the following core service layers:

- ▶ Input, presentation, and output services. These services are leveraged to acquire and ingest content through a variety of input and capture devices and software, generate consistent, well-formed data that can be used directly by another system or displayed through an out-of-the-box or custom user interface to a person, and printed onto a variety of output devices. Examples include capture products, portals, Web clients, and WebDAV (Web-based Distributed Authoring and Versioning) enabled applications.
- ▶ ECM and BPM capabilities. These are high-value business-oriented services that directly leverage the lower level services of the ECM/BPM service bus. These services are unique and make it easy for content and process driven applications to leverage the full scope of functionality in the end-to-end IBM FileNet P8 architecture.
- ▶ ECM/BPM service bus. The services that reside in the ECM/BPM bus represent low-level functionality specifically designed to enable content and process-driven applications. The services can be used to compose more complex composite applications that require significantly less effort than building them from scratch.
- ▶ Data services. The services in this architectural layer are designed to allow insert, extract, and re-use of data from the enterprise data stores and

applications. The layers isolate data consumers from the underlying data constructs and changes. These services provide access to both structured and unstructured data stores and repositories.

- Storage services. These services provide interfaces to physical and virtual storage sub-systems, data management components, and content caching services and are a key component of Information Lifecycle Management (ILM). They abstract out the different capabilities that are provided by different storage vendors into a coherent model.

The IBM FileNet P8 ERA can be broken down into distinct layers, each of which builds upon the others in successive detail:

- Layer 1: Functional areas of ECM capabilities provided by IBM FileNet P8 or through a Partner solution, plus the integration with IT infrastructure and vertical/cross-functional services
- Layer 2: Functional groups
- Layer 3: Key capabilities and services of IBM FileNet P8

Figure 1-3 shows Layer 1 of IBM FileNet P8 Enterprise Reference Architecture.

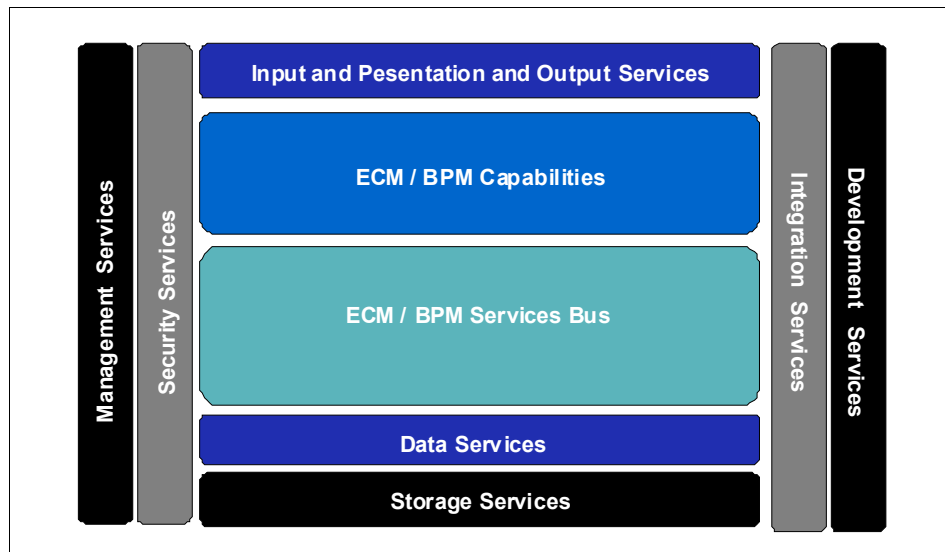


Figure 1-3 Layer 1 of the IBM FileNet P8 Enterprise Reference Architecture

Layers 2 and 3 are not shown in Figure 1-3, owing to the level of detail they incorporate.

Segmenting the IBM FileNet P8 ERA into layers allows discussions involving the Enterprise Architecture to be geared to the appropriate audience, ranging from high-level business strategists to low-level technology architects.

One main advantage of including the IBM FileNet P8 ERA in strategy and design efforts is that it enables a standards-based view of all content, process, and compliance needs of an enterprise, which helps to ensure that future application requirements are addressed in a manner consistent with the overall principles and objectives of the organization.

1.5 Summary

Our goal in this introductory chapter is to provide a basis for understanding the scope of enterprise content management and the resulting need for a platform-based approach to manage the many sources of unstructured data. Also discussed are the various functional capabilities and high-level architecture components of the IBM FileNet P8 Platform along with its resulting ability to scale to handle large numbers of users and vast amounts of data. While an initial discussion of how IBM FileNet P8 integrates into an organization's Enterprise Architecture is provided, the complete IBM FileNet P8 operating environment is presented throughout this book with the aim of demonstrating how IBM FileNet P8 can be leveraged to drive real content- and process-driven solutions.



Core component architecture

IBM FileNet P8 Platform is the unified enterprise foundation for the integrated IBM FileNet P8 products. In this chapter, we describe the core components of IBM FileNet P8 Platform, their architecture, data model, and associated security features.

We cover the following topics:

- ▶ 2.1, “IBM FileNet P8 Platform overview” on page 28
- ▶ 2.2, “Content Engine” on page 29
- ▶ 2.3, “Process Engine” on page 40
- ▶ 2.4, “Application Engine” on page 47
- ▶ 2.5, “IBM FileNet Records Manager” on page 50

2.1 IBM FileNet P8 Platform overview

IBM FileNet P8 Platform is a collection of tightly integrated components that are bundled together under a common platform. The broad functionality provided by these integrated components constitute an enterprise content and process management platform. Some of the key elements of this platform are a content management repository, a process management repository, an out-of-the-box user interface for accessing content and process elements, and a storage framework that can support a wide range of storage devices and platforms.

There are three main IBM FileNet P8 products that comprise the core of IBM FileNet P8 Platform: IBM FileNet Content Manager, IBM FileNet Business Process Manager, and IBM FileNet Records Manager. IBM FileNet Content Manager provides enterprise content management. IBM FileNet Business Process Manager provides business process management. Both of these products are built upon three core components of the IBM FileNet P8 Platform: Content Engine, Process Engine, and Application Engine. IBM FileNet Records Manager is an add-on product that works with IBM FileNet Content Manager and IBM FileNet Business Process Manager to provide compliance management into these products and implemented solutions.

Table 2-1 shows the acronyms that are used for IBM FileNet P8 Platform components.

Table 2-1 Acronyms

Acronym	Term (additional used reference term)
AE	Application Engine (application server)
CE	Content Engine (repository server)
PE	Process Engine (process server)
CSE	Content Search Engine (search component)
RDBMS	Relational Database Management System
FCD	Fixed Content Device (storage device)

Figure 2-1 on page 29 provides a high-level architectural view of these key components and their relative interactions.

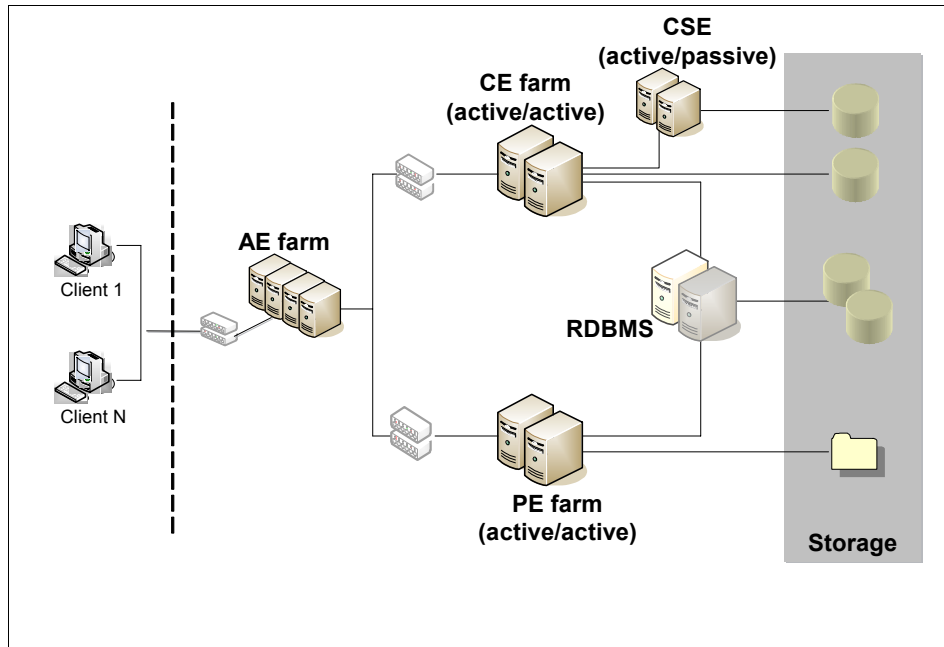


Figure 2-1 IBM FileNet P8 core components

The primary components, which we also refer to as servers (Application Engine, Content Engine, and Process Engine), are delivered as stateless components with farmable architectures that can support both vertical and horizontal scaling. Technology variations in the servers affect the way in which each is load-balanced and provisioned; regardless, each server provides highly performing and highly scalable services in support of handling even the largest enterprise's mission critical loads.

In the remaining sections of this chapter, we provide details about each of the main components: Content Engine (including the storage tier), Process Engine, Application Engine, and IBM FileNet Records Manager (the product).

2.2 Content Engine

The IBM FileNet P8 Content Engine provides an extensible content management platform that supports a wide variety of content-centric applications. The Content Engine is written in Java as a J2EE application, and it is built and deployed as a single EAR file. The EAR file can be deployed in any supported J2EE application server. The IBM FileNet P8 Platform currently supports WebSphere, WebLogic,

and JBoss® application servers and ships with application server-specific EAR files.

The Content Engine application is written based on a generic set of J2EE services and is implemented primarily as a set of POJOs that leverage JDBC™ to access the underlying relational database where all IBM FileNet P8 metadata is stored. These J2EE services are fronted by two stateless EJBs that make up the EJB™ Listener, which demarcates the transaction and authentication boundaries into the server. Various asynchronous activities within the server are managed by a series of background threads.

Figure 2-2 shows the internal system architecture of a Content Engine.

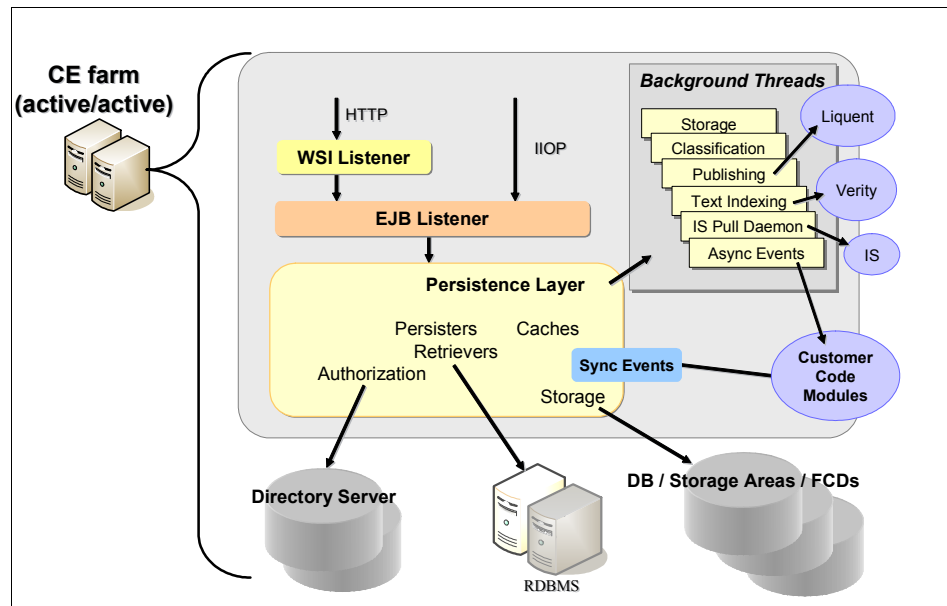


Figure 2-2 Content Engine internal system architecture

Each Content Engine server instance is configured as a member of a given IBM FileNet P8 domain. A Content Engine server instance can support one or more object stores (libraries), where content is stored. Each object store can be scaled to store hundreds of millions of documents and to service requests from thousands of concurrent users. An IBM FileNet P8 domain and its associated Content Engine server instances can support up to 150 object stores.

There is a single global configuration database (GCD) per IBM FileNet P8 domain and there is one to many databases, with one database for each object store, for a Content Engine server instance, as shown in Figure 2-3 on page 31.

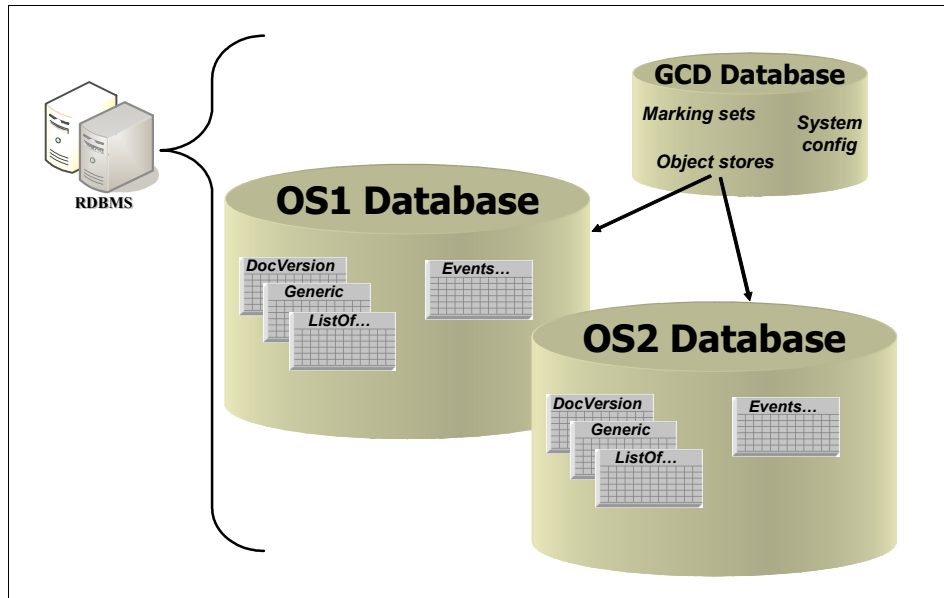


Figure 2-3 Content Engine internal database structure

2.2.1 Data model

Central to the Content Engine is a strongly typed, hierarchical, and extensible object model. The Content Engine comes with a set of defined system classes that you can extend to create custom classes for use within an application. Some examples of the predefined classes are Document, Folder, Custom Object (special object that is content-free with metadata information only), Annotation, and Referential Containment Relationship.

Each Content Engine object has one or more properties that are defined for it. The state of a Content Engine object is exposed and manipulated primarily through the values of these properties. The supported property types include String, Integer, Float, Boolean, Binary, DateTime, ID (GUID), and Object. Properties of type String and Integer can have choice list values associated with them to define the list of acceptable values for that property. Each property has other attributes, such as whether its value can be set, is required, is persistent, and has default values.

A property can be multi-valued or object-valued. Multi-valued properties can be used, for example, to send e-mail to multiple recipients at once. Object-valued properties provide the mechanism by which one object can reference other (one or more) objects within the system, for instance, a Folder object can have an object-valued property named Parent that references the containing object of the

Folder. These references include relationship objects that define one-to-one or one-to-many relationships between objects that can control certain behaviors on those objects.

Content Engine supports versions with major and minor revisions. Document objects can have multiple versions. Custom objects, because they are metadata-only objects, have no associated content elements and are not versionable. Check-out and check-in operators provide facilities for locking a given revision for a given user.

The IBM FileNet P8 data model provides a powerful framework within which you can define complex data models to represent business-specific data and content. In addition, it enables you to operate on and manipulate that data and content in a versioned and consistent manner.

2.2.2 Authentication

Authentication is the process of determining who users are and whether users are who they say that they are. For authentication, Content Engine relies on the J2EE authentication model, which is based on the Java Authentication and Authorization Service (JAAS). Content Engine uses perimeter-based authentication at the application tier to enforce user authentication for all content-related operations.

2.2.3 Authorization

Authorization is the process of determining whether a user is allowed or denied to perform an action on an object. It is managed within the Content Engine by an access control-based authorization model. Individual access rights control which actions can be performed on a given object by a given user or group (known as principals). These individual access rights (called *Access Control Entries or ACEs*), can be grouped together to form an *access control list, or ACL*. By applying one or more ACLs to an object, you can individualize the level of security that is enforced on an object. You can apply ACLs directly to the object, or the object can be secured by ACLs derived from other sources:

- ▶ Default: Each class is created with a Default Instance Security ACL. By default, the Default Security ACL is applied to instances of the class.
- ▶ Templates: Security templates, which contain a predefined list of access rights, can be applied to an object.

- ▶ Inherited: ACLs applied to particular types of folders can be inherited by objects.
- ▶ Proxied: Security settings that are applied to one object can be inherited by another object. The receiving object's original security settings are supplemented, not replaced, by the proxied object's access rights.

Principals that are defined within an ACE for a given ACL represent users or groups that are defined within an underlying LDAP repository. The task of mapping an access right to a user or group is supported by the authorization framework, which manages the user and group look-up in the configured LDAP repository or repositories. Content Engine supports most of the widely used LDAP stores (including Tivoli Directory Server, Active Directory®, and SunOne Directory Server). For more information, refer to IBM FileNet P8 Hardware and Software Requirements.

2.2.4 Event framework

Content Engine provides an extensible framework by which custom code can execute in response to various system-defined events or user-defined events, such as adding a document to an object store. The primary elements of the Content Engine event framework include:

- ▶ Event: A predefined action, such as the creation or deletion of a document.
- ▶ Event action: An object associated with the event, which specifies, through its property settings, which custom code to execute in response to the event.
- ▶ Event action handler: The code, written as custom Java classes that implement the `EventActionHandler` interface.
- ▶ Subscription: An object which, using its properties, specifies one or more events, a target Content Engine object on which those events can be triggered, and an event action object.

When a predefined action takes place on a Document, Folder, or Custom object, an event is triggered and the custom code, which is called the event action handler, is executed. Based on a property setting on the event object, event actions can execute either synchronously or asynchronously. Synchronous events execute within the transaction context of the executing request and can force the overall transaction to fail. Asynchronous events are queued for later processing by the Content Engine server in the background, asynchronous-event thread.

In IBM FileNet P8 4.x, event action handler code is written as custom Java classes that implement the `EventActionHandler` interface. These custom Java classes are delivered as jar files located through the global class path or saved

within the system as special content objects called Code Modules (the jar files are stored as content elements). When a given action occurs on a particular object, a query is executed to find the set of associated subscriptions and their corresponding event action handlers. For each subscription, the event action handler is loaded through a custom classloader and executed through the `EventActionHandler.onEvent()` method.

Event action handlers provide one of the primary ways for customizing Content Engine server-side behavior. It is common to use event handlers to deliver customized behavior for specific events. There is an out-of-the-box event action handler for launching process flows based on a given event action. There is also a `CustomEvent` class, which can be extended to define custom event actions that can be programmatically raised.

2.2.5 Life cycles

Throughout a document's life, it moves from one state to another, such as from Application to Approval in the case of a loan application document. The Content Engine provides document life cycle management through life cycle policies and life cycle actions.

Life cycle policies define the states that a document can transition through. They typically define the actions to occur when a document moves from one state to another. The life cycle policy can also define a security template to be applied to the document when it enters a new state. Life cycle policies can be associated with a document class and applied to subsequent instances of that class or they can be associated with an individual document instance.

Life cycle actions define the action that occurs when a document moves from one state to another. Life cycle actions are associated with state changes in the life cycle policy object.

2.2.6 Storage services

The primary categories of content storage that are associated with the Content Engine are database storage, file storage, and fixed content device storage. These storage options can be used individually or in conjunction with one another. Use IBM FileNet Enterprise Manager to define and configure the storage options. Storage Area objects, created in IBM FileNet Enterprise Manager, associate a storage category with a specific device.

Content is always streamed in chunks from the client to the server. Each chunk can arrive at any server within a farm of servers. The chunks are reassembled on the server and stored in a temporary staging area prior to being committed to the

final destination. The final committal step is different depending on the chosen storage option.

Database storage

Database storage is the mechanism for storing content within the configured relational database management system. Each piece of content is stored as a blob within the content table of the database. After being streamed over from the client in chunks and reassembled, the content is saved into the database as a blob. Content that is streamed to multiple servers in a farm is saved into the database as partial blobs which, when all content is uploaded, are reconstituted as a single blob and stored in the database in a non-finalized state. As part of the metadata committal transaction, this blob is updated to its finalized state.

File storage

File storage represents any device that can be mounted through the Common Internet File System (CIFS) or Network File System (NFS) protocols. The storage device can be direct-attached (either local disk or SAN) or network-attached storage (NAS). All servers in a farmed deployment must have access to all storage areas, which requires that the storage must be shared in some fashion, for example, NAS. Content files are initially written to a temporary file name on the storage device where they ultimately reside. After the file is written, the process is completed in the following order:

1. The metadata updates are committed along with an entry in a content queue table.
2. The entry in the content queue table is processed by an asynchronous background thread in which the content file is renamed to its final name.
3. The entry is removed from the content queue table.

Fixed Content Device (FCD) storage

Fixed content devices represent a variety of IBM and third-party products that deliver additional functionality over standard file systems. This functionality can include Write Once Read Many (WORM), retention management, and others. An example of third-party products includes NetApp® SnapLock®. In addition, IBM FileNet Image Manager repositories can be configured as fixed content devices for Content Engine. Typically, third-party product integrations rely in some part on proprietary device APIs to manage the interactions that are necessary for committing or finalizing the content operations. Much like other storage types, the content is streamed from the client and stored in a temporary staging area on the server. In the same way as for file stores, the content is written to a file storage area and then a subsequent request to migrate the content out of the queue is processed. This migration request triggers the update to the fixed content device (generally through proprietary APIs), after which the system

updates the document entry in the DOCVERSION table with a referral record to the entry in the FCD, and deletes the file in the staging area.

In summary, the stateless, farmable architecture of the Content Engine requires that all storage-related devices must be available as shared resources to all servers in the farm because:

- ▶ Data is streamed in chunks from the client to any server within the farm.
- ▶ Retrieval and committal can be executed by any server within the farm.
- ▶ Every server instance within the farm requires access to the temporary storage area where the content is collected.

Figure 2-4 illustrates the various Content Engine storage services.

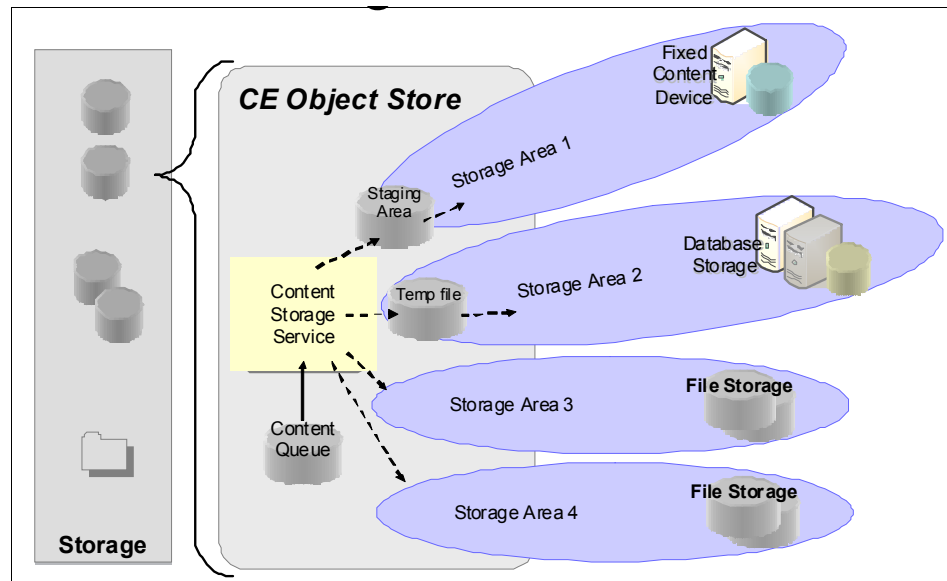


Figure 2-4 Content Engine storage options

2.2.7 Full-text indexing

The Content Engine provides full-text indexing through the Content Search Engine, which consists of the K2 server, which is a search engine from Autonomy. The K2 server provides text indexing and search functionality for a wide variety of document formats. Full-text indexing, and specific metadata attributes, can be configured on a per class basis, for instance, any create or update operation on a given document instance causes the server to evaluate the document's full-text indexing attribute to determine whether to queue the document for indexing.

To handle large indexing requirements, Content Engine can store content into multiple collections, where each collection has its own designated K2 server that performs indexing. The IndexArea object on the Content Engine corresponds to one or more of these collections. An index area is a file system directory that contains the information that is necessary to perform full-text indexing. Using IBM FileNet Enterprise Manager, you can configure 0 to N index areas, either as individual index areas or as index area groups across which we stripe the documents to be indexed.

During the indexing process, the system writes to a single collection. When that collection's capacity is reached, the collection is automatically closed and a new collection opened. The index area in which a collection is stored can be online or offline. If offline, they can be automatically brought online as needed when other index areas reach full capacity.

The Content Engine runs a background service on an indexing table to identify documents that are queued for indexing. The system queries items from the indexing table and submits them, in a configurable-sized batch, to the K2 server. For each entry in the indexing table, the indexing service evaluates both the document to be indexed and the storage medium on which it resides. Because the K2 server does not have access to either the database or to any given fixed content device, documents that are stored in the database or in a fixed content device must be staged to a temporary location on the shared file system before they can be indexed. If a given document has any metadata attributes tagged for indexing, those attributes (name/value pairs) are written to a separate text file that is passed to the K2 server along with the content file(s) to be indexed. Metadata and content files are passed to the K2 server as URIs, which the search engine uses to directly access and read those files for indexing.

Users can run queries against the metadata, the full-text index data, or both. In all cases of a full text index query, the results are dumped into a temporary table and a join is executed across the associated metadata table to handle any metadata-related portions of the query and to provide a means of performing authorization checks on the resulting items. Where multiple, active, index areas are configured for a given class of objects, queries are executed against each index area, and the final result-set is aggregated from them. The final result-set contains only those items for which the calling user has access rights.

Figure 2-5 on page 38 shows the Content Search Engine system architecture.

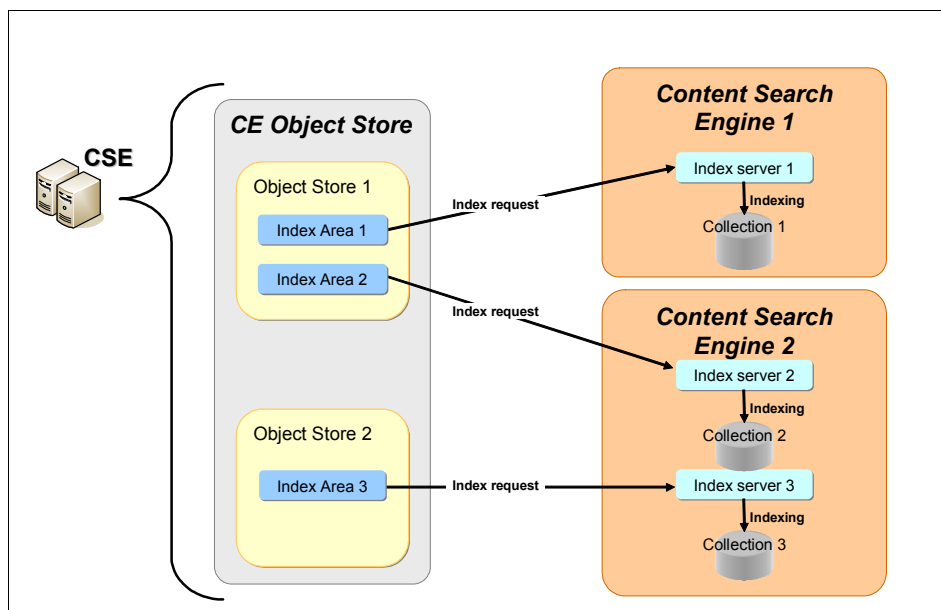


Figure 2-5 Content Search Engine system architecture

2.2.8 Publishing

Content Engine provides publishing services using integration with a third-party rendition engine from Liquent. The rendition engine provides the ability to render various documents' formats into either PDF or HTML. The publishing framework can be leveraged to generate a new version of an existing published document or to generate a new document altogether. The framework supports integration of other transformation services through plug-ins.

The publishing framework consists of two primary components, a publish template and a style template, both of which are defined using IBM FileNet Enterprise Manager. The publish template is used during the publishing operation and defines the properties and security attributes for the published document. The style template defines the output format and various other rendering options, such as watermarks.

2.2.9 Classification

Classification is a general framework that is used within the Content Engine to automatically classify content as it enters the system. Out-of-the-box functionality provided by this framework is the XML classifier, which auto-classifies incoming XML documents.

2.2.10 Protocols

A Content Engine deployment might leverage several protocols. On the client-side, requests come in to the client tier through the Web service listener or the EJB listener. Although the Web services listener communicates through HTTP only, the EJB listener communicates using the same protocol that the application server leverages for its EJB interactions (IIOP, for example). The Application Engine (either Workplace and WorkplaceXT) communicates to back end Content Engine servers through the EJB protocol only. Custom clients can use either the EJB or WSI interfaces. IBM FileNet Enterprise Manager (the Content Engine administration client) uses the .NET API over the Web services (HTTP) interface.

On the server-side, Content Engine uses:

- ▶ JDBC to communicate with the relational database management system, including the database storage areas.
- ▶ NFS or CIFS to communicate with the file storage areas.
- ▶ Various protocols for the fixed content devices. For more information, see the third-party vendor-specific documentation.
- ▶ LDAP to communicate with the underlying directory service provider(s).
- ▶ Liqueur rendition engine's proprietary protocol.
- ▶ Autonomy K2 index server's proprietary protocol.

2.2.11 Tools

IBM FileNet Enterprise Manager is the configuration and administration tool for Content Engine. IBM FileNet Enterprise Manager is a Microsoft Windows application built using the .NET API and communicates with the Content Engine using the Web services interface. IBM FileNet Enterprise Manager supports the following actions:

- ▶ Configuring all aspects of the domain and underlying object stores.
- ▶ Defining custom metadata, such as classes, properties, templates, subscriptions, and event actions.
- ▶ Assigning many aspects of security access rights.
- ▶ Searching for and administering instances of documents, folders, and custom objects.

The J2EE administration console is used for all deployment and application management activities. The standard application server administration console can be used to deploy the content engine EAR file, configure all related

resources (such as data sources, and authentication providers), and start and stop the Content Engine application.

2.3 Process Engine

The Process Engine is a C++ based application that provides an enterprise-wide business process management platform on which to build and deliver enterprise applications. The Process Engine server comprises various processes that execute in a coordinated fashion to manage process requests and execution. The general architecture is a single-threaded, multi-process design in which a multi-threaded broker process accepts all incoming requests and delivers them serially to a series of single-threaded VWKs processes that handle each request and return the results. This thread management occurs invisibly to the user but it is important to understand from a system management perspective. The maximum number of VWKs processes is configurable. Each process starts up as needed to service incoming requests until the maximum number is reached.

A given Process Engine instance is configured as a member of a given IBM FileNet P8 domain. Each Process Engine service is associated with a Content Engine server or cluster through which Process Engine directs all directory server-related calls (for user and group resolution, for instance). A given Process Engine service can be leveraged by one or more Content Engine object stores for managing business processes that are associated with content in those repositories. The corresponding containment model in the Process Engine is called an isolated region. An isolated region contains all of the related process definitions and metadata for a given segment. Unlike Content Engine, however, Process Engine is limited to a small number of isolated regions. The current supported maximum is five. The best practice is to have one isolated region used for a related set of object stores.

Figure 2-6 on page 41 shows the Process Engine system architecture.

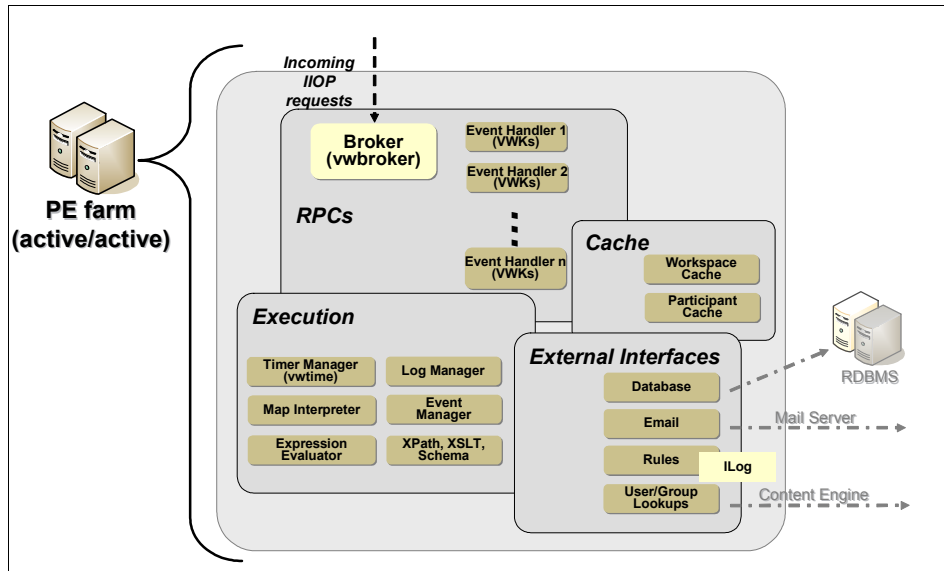


Figure 2-6 The Process Engine system architecture

2.3.1 Data model

The Process Engine has its own database. A Process Engine database is populated by a series of tables representing the three main components of an isolated region: rosters, queues, and event logs.

Each instantiated process flow is represented by one or more objects that represent the current state for each thread (concurrent path) within that process flow. Each of those objects is represented by a single row within a roster table and a queue table. The *roster* represents all of the running instances for a given class of process definitions. You can configure which roster a given process definition is associated with at design time. A *queue* represents all of the running instances that are on a particular step (or related steps) within the process flow. The roster represents objects that are based on their type of process, while a queue represents objects that are based on the type of step they are on, at a particular point in time. Rosters are used to query for objects regardless of where they are in their process flow, typically for administrative purposes, while queues are queried for work to be executed. At any given point in time, each of these objects has one entry in the roster table and one entry in the queue table. Process flows that have multiple, concurrent paths that are being executed have multiple objects that represent each of these paths, each of which have corresponding entries in their respective roster and queue tables. There are also a variety of system tables for managing certain system functions, such as timers and events.

Figure 2-7 shows the tables in the Process Engine database.

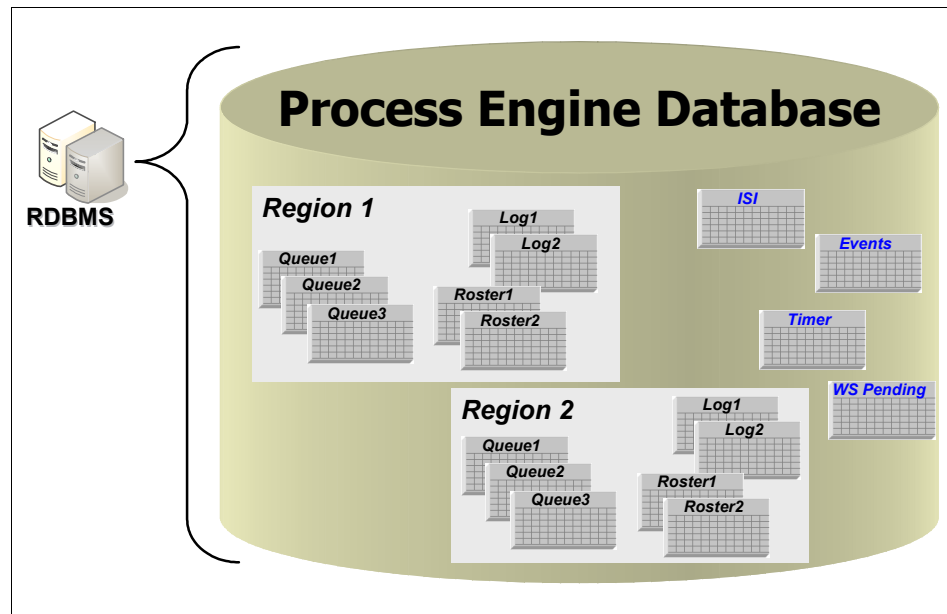


Figure 2-7 Process Engine database

There are different types of queues: process queues, user queues, component queues, and system queues:

- ▶ *Process queues* can be thought of as public queues where numerous users might have access and any of those users are allowed to browse and process work items from those queues on a first-come-first-served basis.
- ▶ *User queues*, in contrast, are associated with a particular user and only have work items that are specifically designated for that user to work.
- ▶ *Component queues* are a special kind of process queue that the Component Manager application uses for background processing of custom actions.
- ▶ There are a number of *system queues* that are used for managing various system activities on work items.

Each work item is represented by an entry in a roster table and a queue table. Each entry is represented by a subset of metadata defined for a given process. Each time an insert or update is done for a given work item, the list of exposed columns on the roster or queue are evaluated (matched by name) and the corresponding set of data from the work item is set. Any data elements that are not specifically exposed are blobbed out into a separate blob column in the associated queue table. What this means fundamentally is that each row in a roster or queue table exposes a subset of metadata that exists within that work

item and everything else is stuffed into the blob column in row in the queue table (see Figure 2-8).

Any given work item can be retrieved and examined through a variety of views. These views are: roster element, queue element, step element, and work object:

- ▶ A *roster element* represents each of the data elements (columns) exposed for a given roster.
- ▶ A *queue element* represents each of the data elements (columns) that are exposed for a given queue, minus the blob column.
- ▶ A *step element* represents each of the data elements defined for a given step in the process flow - these are defined at design time by specifying for any given step what fields are used. Step element data fields might be dynamic in the sense that the definition for a step might include an expression for the data element that is evaluated at runtime when generating the step element.
- ▶ A *work object* represents all of the data elements that are associated with a work item, which includes the exposed fields on the queue and any other data elements that are subsequently stored in the blob field.

Understanding this model is important for performance and scalability reasons. Roster and queue elements are generally the fastest and lightest-weight elements because they represent a subset of the overall metadata and neither requires de-blobbing the blob field to generate. Step elements are usually the next preforming. They do require de-blobbing of the blob field but also often represent a small subset of the overall metadata. Work objects are the most expensive because they require de-blobbing of the blob field, and they return all of the metadata that is associated with a given work item.

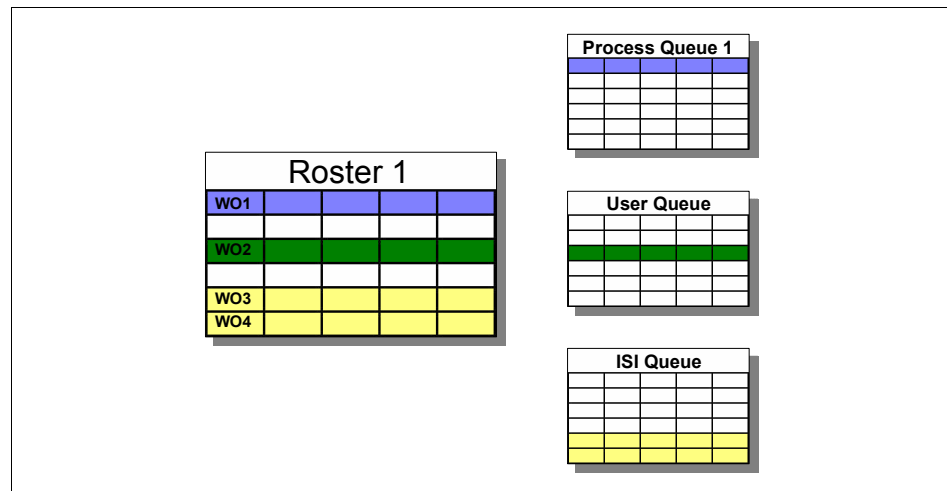


Figure 2-8 Process Engine database schema

2.3.2 Security

Security within the process system is not managed on individual work items but rather on the queues and rosters in which these items reside. User access to work items or portions of work items is controlled by the security rights that they have on the associated rosters and queues that those work items exist in. User queues are additionally enforced by giving access only to the user for which those work items were specifically assigned (based on the current step in the process flow).

2.3.3 Process orchestration

The Process Engine can be either a provider or a consumer of Web services. One mechanism for accomplishing this is through the process orchestration framework. Based on the orchestration portion of the BPEL specification, this framework (shown in Figure 2-9) provides a mechanism by which individual process steps can call out to an external Web service or, conversely, be exposed as a Web service for external consumption.

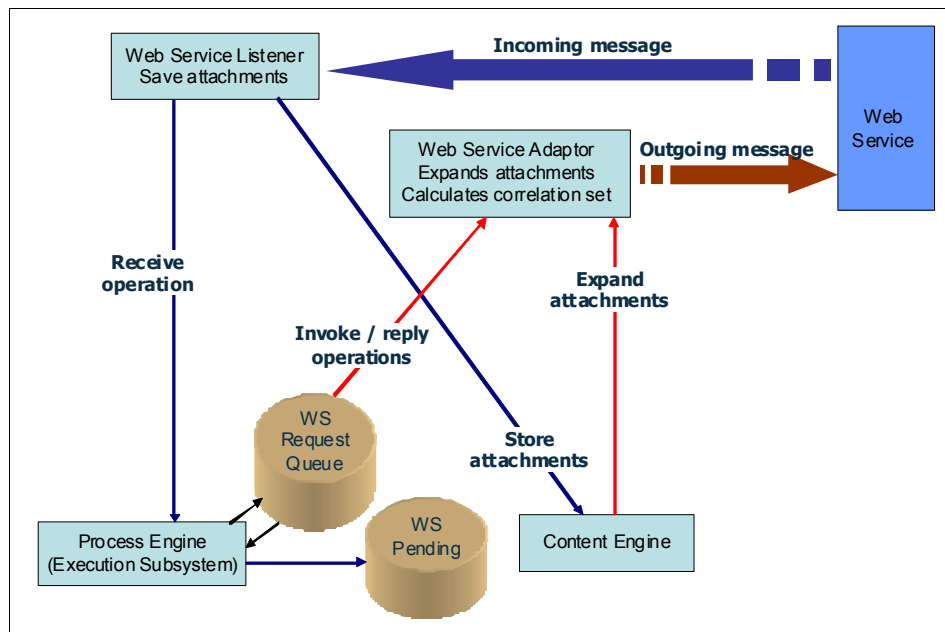


Figure 2-9 Process orchestration

There are three main actions involved: receive, reply, and invoke. The receive and reply steps define a point in the process to expose externally as a Web services entry point and if necessary return a response. The invoke and receive

steps call out to an external Web service and if necessary receive a subsequent response. Figure 2-10 depicts the various interactions that are enabled through the process orchestration framework.

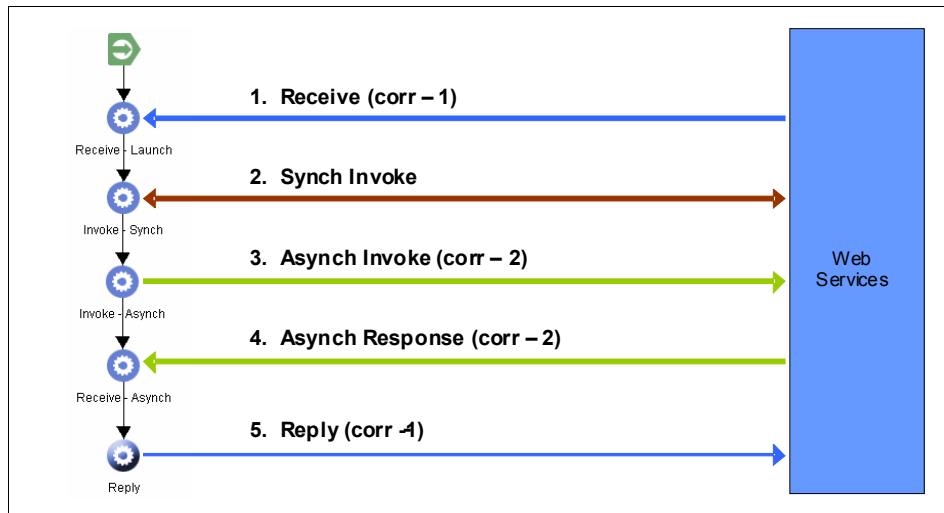


Figure 2-10 Process orchestration interaction

2.3.4 Event logging

Event logs are used to log event data for various activities that occur on work items within the system. Like rosters, there is a default event log and additional event logs can be configured. Process definitions can be associated with a given event log such that subsequent actions that occur on instances of those process definitions can be logged. Event logs, like rosters, are separate tables within the Process Engine database. Each row represents an event action for a specific work item. Also like rosters (and queues), the metadata that is collected in these tables is defined by the columns that are configured for a given event log. The column name and type are matched with corresponding properties of a work item. For those that match, the data is populated from the work item. The types of events that get logged are configurable and provide a fair degree of control over how much data is collected and what it represents. The event log table can be queried to retrieve data on historical events that occurred within the system. The Tracker application uses this data to show the history for a given work item. The event log tables are also the primary source of data for the analytics engine.

2.3.5 Analytics

The Analytics Engine monitors the event log tables and extracts historical process data and pushes it into a series of OLAP cubes for subsequent analysis and reporting functions. Currently, Microsoft SQL Services is the underlying OLAP repository for storing that data. Various OLAP clients can be used for viewing and analyzing that data including simply using Microsoft Excel® to interface with those cubes.

2.3.6 Monitoring

The Process Engine has a number of integrations for providing monitoring. IBM FileNet Business Activity Monitor works off of the event log to monitor events within the system. It provides extensive functionality for managing key performance indicators and reacting to extraordinary conditions. There is also integration with the WebSphere Monitor tool. Process Engine generates common base events, which are what the WebSphere monitor tool consumes and manages.

2.3.7 Rules framework

The Process Engine includes a rules integration framework through which it interfaces with third-party rules engines. The Process Engine can integrate with a number of key rules engines out-of-box. It also provides a framework for additional integrations. Using rules engine integration, you can define points in the process where you link to an associated rules engine to evaluate certain data or provide data from various defined rule sets that can be used to make subsequent process flow decisions or be included as data to users for processing steps within the flow. The rules engines provide one more dimension to the dynamic process flow models by allowing runtime evaluation of attributes or dynamic lookups of data for use within the process flows.

2.3.8 Protocols

There are a variety of protocols potentially being leveraged within a Process Engine deployment. Starting at the client tier, requests might come in through various out-of-the-box applications that make SOAP requests over HTTP to the Application Engine server. Custom applications that leverage the Java API communicate over IIOP to the Process Engine server. Lastly, there are a set of exposed Web services that can be accessed through HTTP.

After it is in the server, the Process Engine uses the protocols:

- ▶ Various database specific protocols depending on the database
- ▶ Web services calls over HTTP to Content Engine to resolve users and groups

2.3.9 Tools

There are many tools that the Process Engine provides for designing processes, administering process instances, configuring the data model, administering the server functions, and doing low level analysis of various server activities. The Process Designer tool provides the general process design capabilities where users (typical business and IT analysts) define their process flows. The Process Administrator tool lets an administrative user query the system for process instances and view the current state of those instances. The Process Tracker tool can be launched to view the current and historical state of an individual process instance. The Process Configuration tool defines the rosters, queues, event logs, and various other system-related components. There is also a Task Manager tool that is deployed with the Application Engine that can be used to start and stop the various server components, including the server itself. There are a series of lower-level tools on the server (such as vwtool and vwspy) that can be used for viewing detailed information about server state and activities that occur there.

2.4 Application Engine

The Application Engine represents the out-of-the-box user interface for the IBM FileNet P8 Platform through its rendering of the application called WorkplaceXT (formerly Workplace). WorkplaceXT is a DOJO-based application that is deployed within the Web container of various supported application servers. It has a variety of heritage JSP™ and Servlet components. Workplace provides a general folder-based view of an IBM FileNet P8 content repository along with various process manager components for representing information, such as inboxes, public queues, and step processors.

WorkplaceXT allows a variety of customization and extensibility options and allows various user preferences to be set to control behavior and presentation. WorkplaceXT also provides a role framework for defining various roles that are associated with content and process-related activities to define what a user can see and do based on their role.

The Application Engine also represents the container under which many of the other product extensions (such as IBM FileNet Records Manager and IBM FileNet eForms) are deployed. It represents the container under which many

customers deploy their custom applications that leverage the underlying process and Content Engines. Because the Application Engine represents the entry point of users into the system, it is at this tier where all of the authentication activities are generally handled. The IBM FileNet P8 4.x authentication model is based on the J2EE Java Authentication and Authorization Service (JAAS) and assumes perimeter-based authentication that is managed by the application servers, which are configured authentication providers.

WorkplaceXT, IBM FileNet Records Manager, eForms, and other IBM FileNet P8-based applications generally leverage a combination of the IBM FileNet P8 content and process Java APIs. By default, the content Java API leverage the EJB interface for its interactions with the server, which leverage whatever protocol the application server supports for EJB interactions (for example, IIOP and T3). The process Java API leverages IIOP for its communication with the process engine server.

There are five main areas within WorkplaceXT:

- ▶ **Tasks:** Provides views into inboxes and public queues for working on business process work items.
- ▶ **Browse:** A general folder browse interface for browsing through documents stored in folders.
- ▶ **Search:** Access to various search facilities for locating documents within the system.
- ▶ **Author:** Hosts authoring tools related to the content and process services.
- ▶ **Admin:** Hosts administration tools related to the content and process services.

Figure 2-11 on page 49 illustrates the Application Engine system architecture.

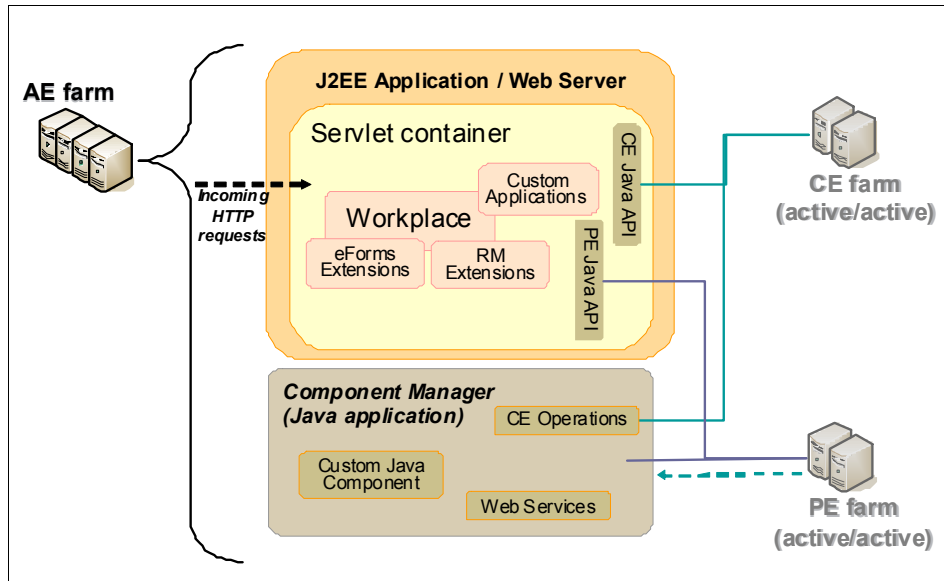


Figure 2-11 Application Engine system architecture

2.4.1 User preferences

User preferences are a collection of configuration settings that allow a user to modify various display and functional characteristics of WorkplaceXT. Those preferences are stored within a user preference object in the Content Engine repository and retrieved at logon to customize the user experience.

2.4.2 Microsoft Office and Outlook integration

Application Engine also provides a Microsoft Outlook and Microsoft Office integration facility that enables basic IBM FileNet P8 content management facilities integrated within your Microsoft Outlook and Microsoft Office applications. The Microsoft Outlook integration allows a user to browse for a specific document and include that document as an attachment or link. It can also be configured to save certain e-mails as documents in an associated IBM FileNet P8 object store and, optionally, to delete that e-mail after it is saved. The Microsoft Office integration provides the ability to add documents to an IBM FileNet P8 object store, to browse or search for documents within an IBM FileNet P8 object store, and to open documents within an associated Microsoft Office application. Modified documents can subsequently be checked back in to their respective IBM FileNet P8 object store.

2.4.3 Component Manager

The Component Manager is a Process Engine component, but it is hosted and managed on Application Engine. The Component Manager provides an integration framework that enables the Process Engine to make calls on external components. These might be Java components, Web services, or JMS queues, for example, the Component Manager includes a Web services adaptor that handles outbound Web services calls for the process orchestration framework. It also handles calls on custom Java components that can be registered within the system and called at explicit points in a process. The Component Manager also hosts the CEOperations component, which can be used for executing certain content-related operations against the Content Engine.

2.5 IBM FileNet Records Manager

The IBM FileNet Records Manager (RM) is an add-on product to the IBM FileNet Content Manager suite. It allows enterprises to classify, apply retention policies to, and store records according to fiscal, legal, and regulatory requirements, supporting the entire life cycle of documents or physical artifacts, which can be managed by the software from a single administrative module. RM is DoD 5015.2-STD Chapter 2 (mandatory requirements) and 4 (classified records) certified.

Records can be declared manually or automatically through event subscriptions or workflows. Other applications, such as IBM Content Collector for Email, can also declare records automatically as documents are ingested. Holds are typically placed on records that are identified during discovery as potentially relevant to a legal proceeding. If a record is held, then it is not destroyed when its retention period ends. In the context of RM, a record is a file with metadata that references and contains information about an actual electronic file (document) or a physical object that is classified as record. You create a record to place the document or physical object under corporate or governmental control, which specifies how the document or object is to be stored, accessed, and eventually disposed of. The metadata is in the form of record properties, such as media type, format, author, subject, reviewer, location, and publication date.

RM is a Java application that is installed on top of Application Engine, although there are some administrative tools that are run from the command line. Records can also be declared from within applications, such as Microsoft Outlook, Word, and Excel. RM leverages features and storage capabilities of the Content Engine, so RM does not require a separate database or other storage area for its data. For electronic records that are stored in the Content Engine, RM manages

the security of the record object so that it cannot be deleted until its retention period ends and it is not under any holds.

Figure 2-12 shows some of the major IBM FileNet Records Manager components within the IBM FileNet P8 architecture and their relationship to the underlying core IBM FileNet P8 Platform services.

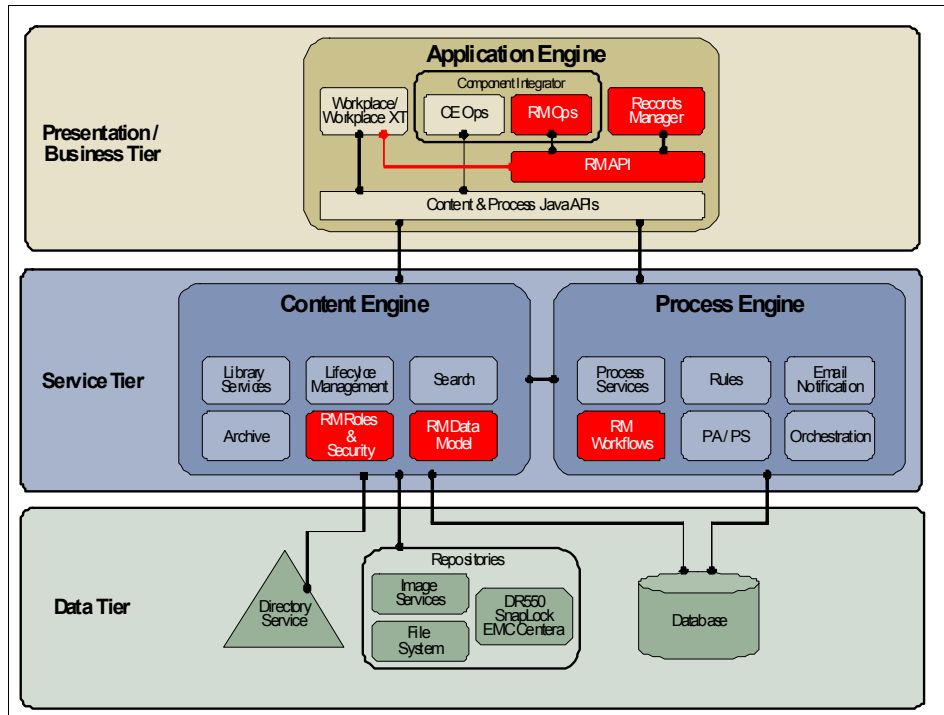


Figure 2-12 IBM FileNet Records Manager architecture as an extension of the IBM FileNet P8 Platform

2.5.1 Data model

The Records Manager implements its entire data model as objects within Content Engine. If electronic records are being managed, the object store where they reside is known as a Records-enabled Object Store (ROS). For each physical or electronic record being managed, a Record Information Object (RIO) is created. A special object store, known as the File Plan Object Store (FPOS), is used to store these objects with the Record Categories, Folders, Volumes, and Information Objects that are represented by Content Engine folders and objects. A single FPOS can support one or more file plans, and it can correspond to zero (in the case of physical records), one, or many different Records-enabled Object

Stores. A single IBM FileNet P8 domain can have multiple File Plan Object Stores.

A Record Information Object provides metadata about a document or physical object that is placed under the control of the RM application. A record might inherit some of its behavior from the record folder in which it is created, for example, it might inherit the disposition schedule of the parent record folder. Records can be categorized as Electronic, Marker, Vital, or Permanent records. Records that have common features can be associated with a record type, for example, you might define a record type to specify a common disposition schedule for records and to aid the search and retrieval of records. You use record types when a group of records that exist under a record folder should have a disposition schedule that is different from the one that is currently associated with the record folder.

2.5.2 File plans

For records management, the Records Administrator creates one or more file plan data models. The data models include Base, Department of Defense (DoD), and Public Records Office (PRO). File plans are hierarchically structured, as shown in Figure 2-13 on page 53, into (potentially) Record Categories, Record Folders, and Record Volumes. Records are cataloged under these schemes based on business functions. Retention schedules can be defined, and records can be reviewed before they are destroyed.

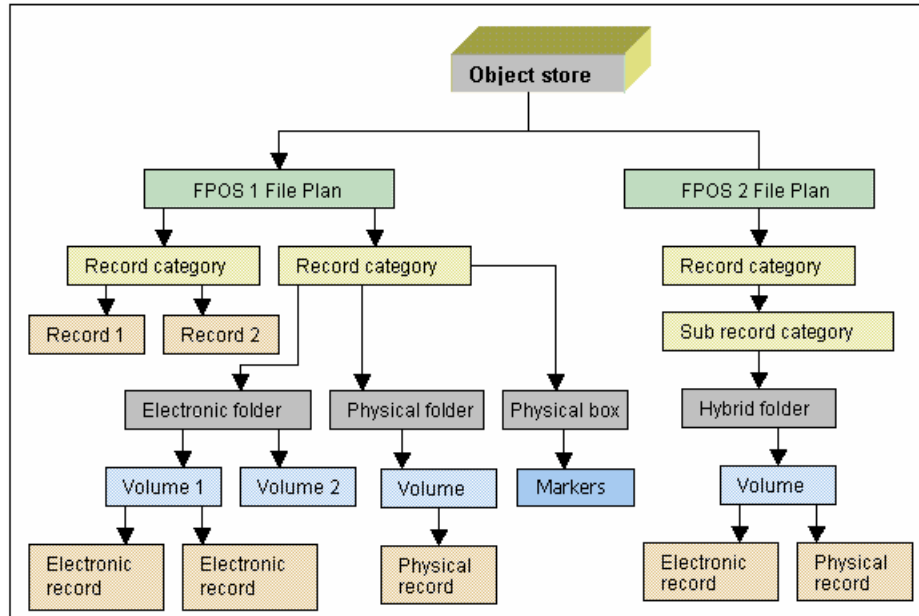


Figure 2-13 File plan structure

Record Categories maintain a set of related records within a file plan. They are created to catalog records based on functional categories. A record category can contain subcategories or record folders (but not both), depending upon which data model is in force. Record Folders serve as a container for related records. They manage records according to the specified retention periods and disposition events. You can create electronic, physical, and hybrid record folders under a category to manage electronic and physical records.

Record Volumes are a logical sub-division of a record folder into smaller and easy-to-manage units. A volume has no existence independent of the folder. A record folder always contains at least one volume, which is automatically created by the system when a record folder is created. Thereafter, you can create any number of volumes within a record folder.

A file plan determines the security and disposition of records. By default, child entities inherit the security and disposition schedule of their parent container. In the case of electronic records, the security on the document object is changed to that of the Record Information Object so that the document object cannot simply be deleted. A declared document cannot be deleted until its associated Record Information Object is deleted. The constraint of deleting a document is imposed by a property on the document that points to the RIO and uses the Prevent Delete action. A user with Full Control access rights cannot delete a declared

document. The system automatically deletes a document when the document's associated record is deleted. The delete action occurs because the object-valued property on the record points to the document and uses the Cascade Delete action.

Besides providing a logical partitioning of the Record Information Objects, the classification hierarchy allows you to aggregate records under different folders within a classification scheme based on business functions. Aggregation of records within RM is important. During disposition processing, if aggregating at the container level, only the retention criteria of the container must be examined, which is potentially much faster than examining criteria on each individual RIO within that container. On the other hand, all of the records within such a container are held, even though perhaps only a small subset of them is actually needed. When the container holds a significant volume of records, holding many records unnecessarily might not be acceptable in all business environments.

2.5.3 Additional references

For more information about the Content Engine, Process Engine, Application Engine, and IBM FileNet Records Manager, refer to the respective product manuals and the following IBM Redbooks publications:

- ▶ *IBM FileNet Content Manager Implementation Best Practices and Recommendations*, SG24-7548
- ▶ *Introducing IBM FileNet Business Process Manager*, SG24-7509
- ▶ *Understanding IBM FileNet Records*, SG24-7667



Expansion products for content ingestion

In addition to the strengths of the core IBM FileNet P8 Platform, IBM FileNet P8 provides a number of major expansion products that extend enterprise content management functionality. The expansion products implement critical features to ingest, organize, and access content, build applications and automate processes, and support discovery and compliance. In this chapter, we provide an overview of the expansion products and specifically discuss the content ingestion related expansion products. The other expansion products are discussed in the three chapters following this one.

We cover the following topics in this chapter:

- ▶ 3.1, “Expansion product overview” on page 56
- ▶ 3.2, “Content ingestion products overview” on page 57
- ▶ 3.3, “IBM Content Collector” on page 58
- ▶ 3.4, “IBM FileNet Capture” on page 67
- ▶ 3.5, “Summary” on page 80

3.1 Expansion product overview

IBM FileNet P8 expansion products facilitate getting content into IBM FileNet P8, accessing content outside of IBM FileNet P8, building applications for IBM FileNet P8, and supporting discovery and compliance.

Companies constantly receive content from external sources and create new content. Products that automate content ingestion while making it relevant and accessible are key for content management. The IBM FileNet P8 expansion products for content ingestion take paper, faxes, e-mails, and other forms of information and organize it and insert it into IBM FileNet P8. Content that is already available in other repositories and locations can be federated or fed automatically into the system with connectors and federation products.

After the content is available and organized, tools that automate processing the content and making it *active content* enables the organization to revise and optimize their business. Optimization and analysis products enable managers to respond, predict, and streamline their business processes.

IBM FileNet P8 Platform serves as a foundation on which applications can be integrated, built, and deployed. IBM FileNet P8 architecture provides the framework for rapidly developing functionality. IBM FileNet P8 provides a number of expansion products that demonstrate the accelerated application deployment and lower total cost of ownership using this architecture. These applications support the core tenant of the information agenda: Content anywhere.

Finally, it is necessary to secure, maintain, preserve, and retain the content in a useful and controlled manner. Legal requirements for document life cycles and discovery drive the IBM FileNet P8 expansion products in this area, which help businesses find and keep the right information at the right time. These same requirements also help to expand document usefulness by annotating and classifying them for reuse. Additionally, these products reduce duplication and maximize storage to reduce the total IT budget.

In this book, we categorize the expansion products as:

- ▶ Content ingestion products:
 - IBM Content Collector
 - IBM FileNet Capture
- ▶ Connectors and federation products:
 - IBM FileNet Application Connector for SAP R/3
 - IBM Content Integrator
 - Content Management Interoperability Services
 - IBM FileNet Services for Lotus Quickr

- IBM FileNet Connectors for Microsoft SharePoint
- IBM FileNet Content Federation Services (not covered in this book)
- ▶ Application frameworks products:
 - Electronic forms (eForms)
 - Business Process Framework
 - Business Activity Monitor and Cognos Now
- ▶ Search, classification, and discovery products:
 - IBM Classification Module
 - IBM Content Analyzer
 - eDiscovery Manager and Analyzer
 - IBM OmniFind Enterprise Edition

In this chapter, we discuss content ingestion related expansion products. In the next three chapters that follow this chapter, we discuss products that provide connectors and federation, application frameworks, and support for discovery and compliance.

Note: IBM Content Analyzer (formerly known as OmniFind Analytics Edition) and IBM FileNet Content Federation Services are not covered in this book. They are covered in the following IBM Redbooks publications:

- ▶ *Introducing OmniFind Analytics Edition: Customizing Text Analytics*, SG24-7568
- ▶ *Content Federation and Integration* - To be published in 2010

3.2 Content ingestion products overview

Very few companies start their IBM FileNet P8 systems in a vacuum. Typically, you have existing and new content that you want to utilize. The content ingestion products provide a solution to pull in documents from file systems, e-mail servers, incoming faxes and images, and paper sources into IBM FileNet P8 system. These products not only collect this information and centralize them in the IBM FileNet P8 platform, they provide an abundance of other functionality.

Three key expansion products that focus on content ingestion are:

- ▶ IBM Content Collector (ICC): Expands and consolidates the previous functionality of IBM CommonStore, IBM FileNet Email Manager, and IBM FileNet Records Crawler, into a more flexible software.
- ▶ IBM FileNet Capture Professional: A veteran product in the IBM FileNet portfolio that customizes high-volume document capture functionality to fulfill enterprise requirements.
- ▶ IBM FileNet Capture Advanced Document Recognition (ADR): A veteran product in the IBM FileNet portfolio that customizes high-volume document capture functionality to fulfill enterprise requirements.

3.3 IBM Content Collector

IBM Content Collector (ICC) consists of two offerings:

- ▶ IBM Content Collector for Email
- ▶ IBM Content Collector for File Systems

The offerings are part of a family of IBM ECM Content Collection and Archiving offerings. One of the key features of these offerings is that they are completely integrated into the IBM FileNet P8 Platform. Both are similar, using a rules-based connection framework that simplifies and automates the process of collecting, enhancing, and managing content. ICC for Email collects e-mail from a variety of sources. It addresses four main use cases: storage space management, compliance and legal obligations, knowledge extraction, and using e-mail as part of a business process. ICC for File Systems, which collects documents from NTFS file systems, perform similar functions.

In both offerings, ICC reads the sources and applies rules to decide if and how the messages and their attachments are processed and where they are stored. Additionally, many pre and post-processing options, such as classification, the replacement of content with links to the object store, and de-duplication (for e-mails) can be utilized. Because both offerings are almost identical except for the source of the content, we discuss them together in this section. We highlight the differences when appropriate.

3.3.1 IBM Content Collector overview

ICC provides a Configuration Manager for designing *task routes*. Task routes specify how and where the content is collected and processed within the system. Additionally, ICC works with other products for content classification and records declaration. ICC also provides a Web application for e-mail search and retrieval.

ICC Configuration Manager

The ICC Configuration Manager is the administration interface for all configuration that is required for the ICC Archiving Engine. Other administrative tasks for the repository are managed using their respective administration tools. The interface is a workflow-like designer based on IBM FileNet Email Manager 4.0. These workflows are called task routes, which detail flexible e-mail and document capture and processing options using conditional rules that determine if and how to capture content. Several template task routes for ICC Email are installed with the product. Figure 3-1 shows the IBM Content Collector Configuration Manager with a sample task route.

ICC Configuration Manager also supports rule-based journaling archive functions for MicroSoft Exchange and IBM Lotus Notes e-mails.

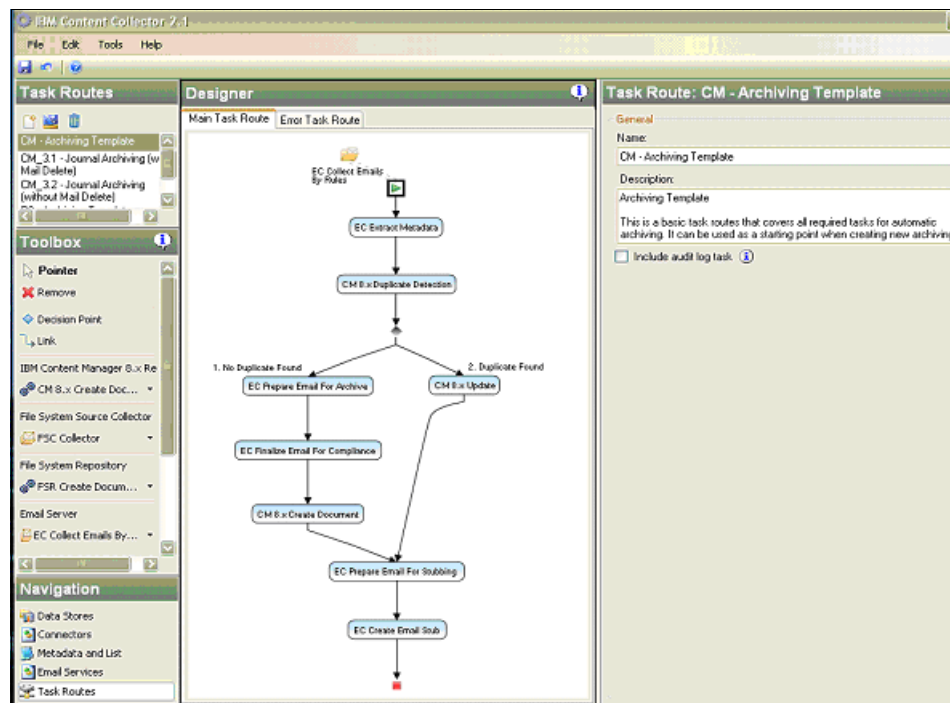


Figure 3-1 IBM Content Collector Configuration Manager with a sample task route

Task routes

A task route is a visual representation of the route that content (such as e-mails, attachments, or files) goes through in the system, from being collected to being stored in the repository. Task routes enable users to apply rules at multiple points in the capture process using a decision point. *Decision points* allow

conditional processing of e-mail and files using rules with potentially different outcomes for each.

Processing e-mails and files include extracting metadata, classification, extracting content, and using metadata to support de-duplication functions for e-mails. This is also where stubbing (for emails) is managed, including stub life cycle and repository organization and location for the content.

Task connectors are used with task routings to provide expanded and flexible functions for the information that is collected.

Content classification and records declaration

IBM Content Collector can call IBM Classification Module to derive classification and metadata information. IBM Content Collector uses task routes to support flexible classification. By organizing data that is to be stored in IBM FileNet P8, the information becomes more accessible, useful, and flexible.

In addition to utilizing IBM Classification Module to classify the collected content, IBM Content Collector integrates with IBM FileNet Records Manager to support automatic declaration of e-mail and files as records during the content capture process.

ICC Web application

The ICC Web application is used for viewing e-mails and searching e-mails in a Web browser. The actions can be triggered directly from the users' e-mail clients and the e-mail authentication controls security, which provides support for SSO. Users do not have to authenticate against the IBM FileNet P8 system. WebSphere Web Application Server 6.1 is installed by default.

Figure 3-2 on page 61 shows the user interface for the e-mail search Web application.

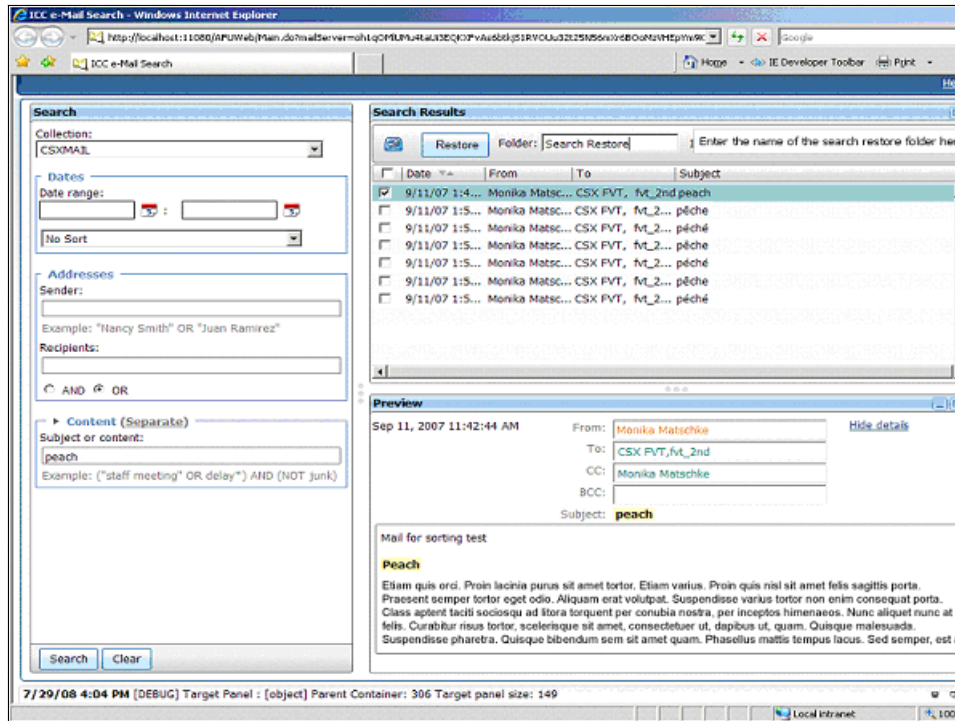


Figure 3-2 ICC e-mail search Web application

In this Web application, the user searches against their collection and can preview the results. Search texts are highlighted in the results window, which is very similar to eDiscovery searches.

3.3.2 System architecture

IBM Content Collector resides on a Microsoft Windows server and sits between the e-mail servers or file systems and the designated content repository. IBM Content Collector for Email supports Microsoft Exchange and Microsoft Outlook 2003 and Microsoft Outlook 2007, including PSTs, IBM Lotus Domino, and Notes 7.x and 8.x. IBM Content Collector for File Systems supports NTFS file systems.

IBM Content Collector supports content archiving of the following target repositories:

- ▶ IBM FileNet Content Manager 4.0+
- ▶ IBM Content Manager 8.3+ on Windows/AIX/Solaris™
- ▶ IBM FileNet Image Services 4.x (File System only)

IBM Content Collector supports read-only function for the following systems:

- ▶ IBM Content Manager OnDemand 8.3+
- ▶ IBM Tivoli Storage Manager

E-mail is archived in a format that supports for legal discovery, storage management, and duplication management. To support the use of e-mail and attachments in business processes, e-mail from Microsoft's Exchange can be stored in MSG format, and messages from Lotus Domino can be stored as CNS format. For both of these formats, e-mail can be embedded or separated. All e-mail can also be stored in .TXT format. Attachments are always separated in this case.

The configuration database can use a repository database (IBM FileNet Content Manager or IBM Content Manager) rather than a separate database for configuration management information.

Note: If NTFS access-only is needed, and a separate database (DB2®, Oracle® or SQL Server®) is required. See product documentation for details of version support and compatibility.

There are five main components of IBM Content Collector from a system architecture perspective:

- ▶ Archive Engine
- ▶ Connectors
- ▶ Repository and back end
- ▶ Web applications
- ▶ E-mail client

From an administrator's point-of-view, there is an additional component, the ICC Configuration Manager, which we mentioned earlier in "ICC Configuration Manager" on page 59.

Figure 3-3 on page 63 illustrates the IBM Content Collector system architecture.

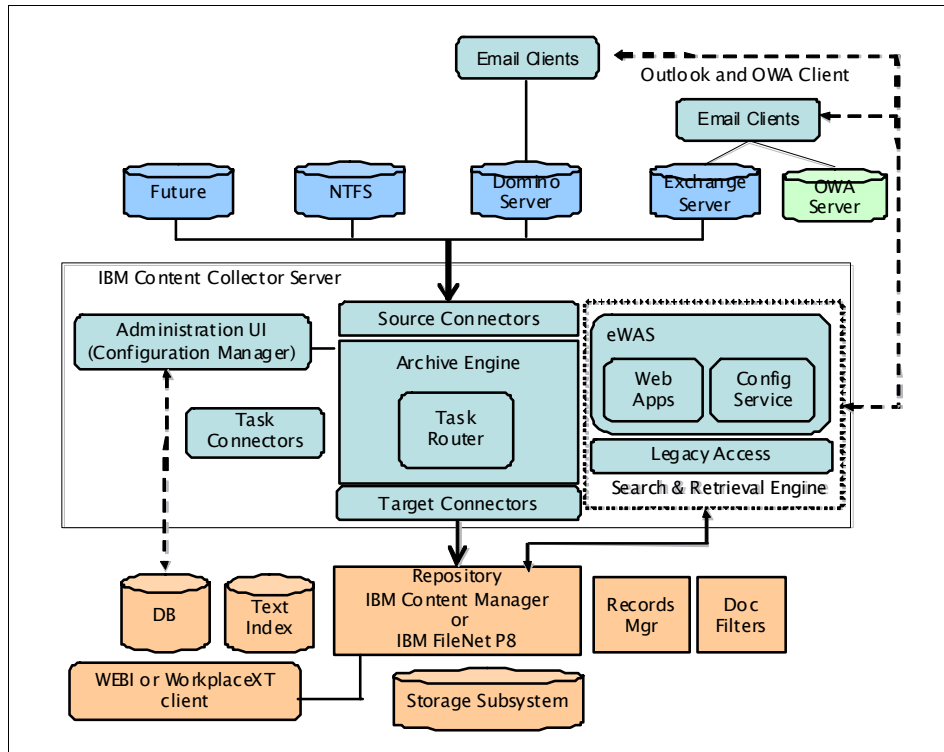


Figure 3-3 IBM Content Collector system architecture

In Figure 3-3, the system architecture diagram, the IBM Content Collector box contains all of the items that are installed on the ICC Server: Archive Engine, the connectors (Source and Target Connectors), and the Web application. E-mail servers and NTFS systems connect through these connectors. The Notes, iNotes®, Microsoft Outlook, and Microsoft Outlook Web Access users can connect directly to the e-mail Search and Retrieval engine to retrieve those documents. Text indexing is performed by Verity (in IBM FileNet P8) or NSE (in IBM Content Manager) and is required for searching from the e-mail client, cross-mailbox searching, or legal discovery using eDiscovery Manager. Target connectors connect the external repositories to ICC.

Archive Engine

Figure 3-4 on page 64 illustrates the Archive Engine architecture.

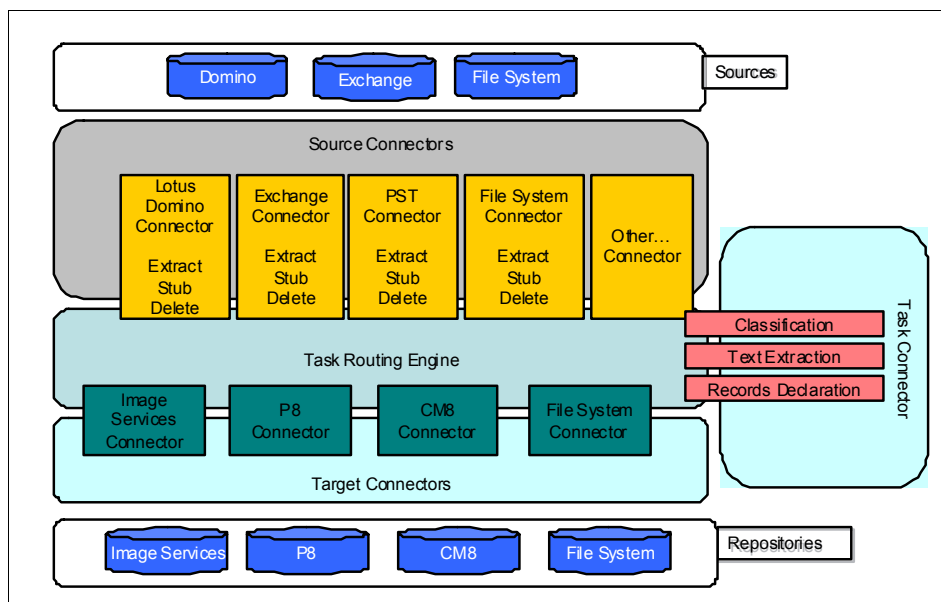


Figure 3-4 IBM Content Collector Archive Engine

The Archive Engine provides the hub into which source connectors, task routes, and target connectors are plugged into an API and contains the business logic to take content from an input source and archive it in an output destination based upon a series of defined task routes (processing rules). Multiple task routes can be processed simultaneously.

Connectors

Three sets of services run on the Content Collector server:

- ▶ Source connectors: Lotus Domino Connector, Microsoft Exchange Connector, PST Connector, File System Connector, and others connectors
- ▶ Target connectors: Image Services Connector, P8 Connector, CM8 Connector, and File System Connector
- ▶ Task connectors: Classification, Text Extraction, and Records Declaration

A single Content Collector server can connect to multiple types of source e-mail systems, file systems, and multiple IBM FileNet Content Manager (P8) or IBM Content Manager (CM8) repositories. All of these services use the same APIs as part of the modular architecture.

In a retrieval or discovery architecture, eDiscovery manager uses the same Web services that e-mail client use to retrieve e-mail from the repositories.

3.3.3 Connection and integration points

IBM Content Collector provides a tight integration with the IBM FileNet P8 repository and IBM Content Manager repositories. ICC can automatically capture e-mail from a monitored mailbox or a file system to an IBM FileNet P8 repository and initiate a manual or automatic business process based on that event.

To connect to source e-mail servers, ICC must be connected to the e-mail server over a network, which requires the Content Collector server to be in the same domain as the source e-mail server or in a separate domain that has a trusted relationship with the domain in which the source e-mail servers reside. Content Collector also requires a single administrative account on each source e-mail server to facilitate a connection and enable Content Collector to take actions on the server.

Application integration

IBM Content Collector can be integrated into Lotus Notes, Lotus iNotes, Microsoft Outlook, and Microsoft Outlook Web clients. The optional outlook extension for Microsoft Outlook and Lotus Notes template modifications are available and can be installed on users' desktop computers to allow for the use of advanced shortcuts (stubs) so that users only need to click an e-mail shortcut in their inbox to open an e-mail in the repository. Microsoft Outlook Web Access (OWA) requires a separate installation on the OWA server.

ICC supports offline access for both Microsoft Outlook and Lotus Notes. It seamlessly retrieves e-mail from the local storage for stubbed mail, which works through the OST (Microsoft Outlook) or local replica (Lotus Notes) replication process, and is synchronized automatically with the repository. Users install this package and determine cache size and deletion policies when the cache is full.

Microsoft Outlook and OWA

In Microsoft Outlook, a toolbar and a menu for ICC allows users to access ICC functionality, which includes archive, search, restore, and mark for stubbing. These features can be selectively installed. The menu also enables them to configure behavior when ICC takes action, such as configuring the location for archiving.

Integration for Microsoft Outlook Web Access is also through buttons in the toolbar. Configuration for OWA functionality for ICC is managed in the ICC Configuration Manager. Administrators can enable and disable features centrally.

Lotus Notes and iNotes

In Lotus Notes, the ICC toolbar is under the Action menu, as shown in Figure 3-5, while Lotus Domino Web Access has a menu just for IBM Content Collector. Notes and iNotes users get ICC functions through template modifications.

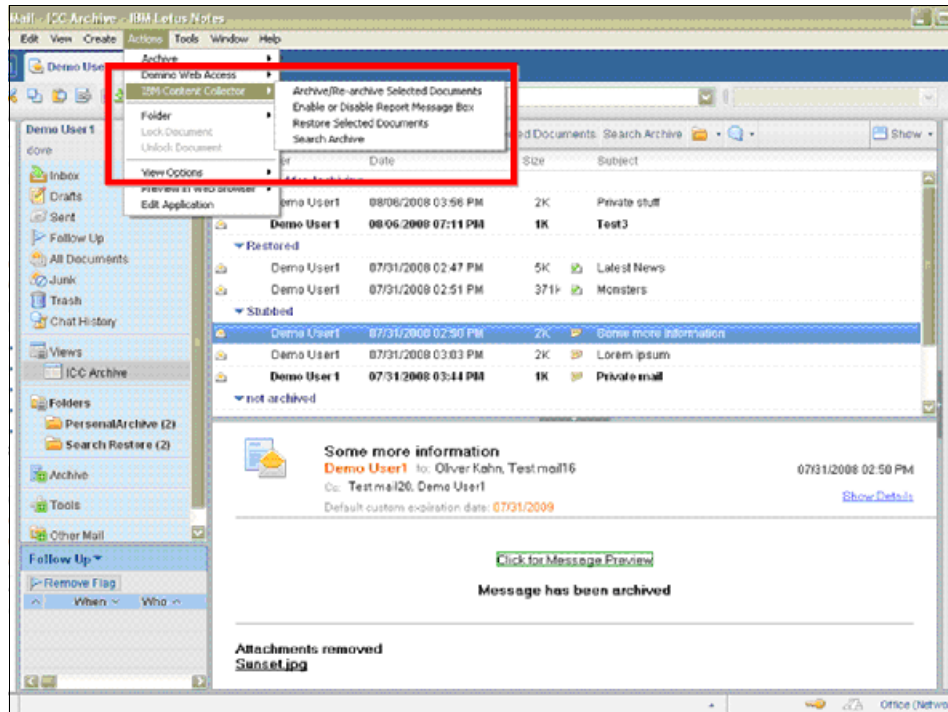


Figure 3-5 ICC toolbar in Lotus Notes

3.3.4 ICC summary

IBM Content Collector enables organizations to take back control and unlock business value of content while enforcing compliance and operational policies, all with lowest total cost of ownership. With automation and centralized administration, compliance requirements become easy to address. Using classification and metadata makes the information more useful, and reusable, which enables businesses to extract more information out of their content. And finally, ICC can start processes and workflows as e-mail and files are incorporated into the system.

ICC provides a modular, extensible architecture for collecting content from multiple sources and providing a means of applying flexible rules to the

disposition of the content. The growing need to manage e-mail makes the ability to enhance and manage e-mail increasingly critical. Organizations can use these tools to reap the benefits of IBM FileNet P8 *active content* concepts quickly, providing increased organizational agility, lower costs, and better compliance at the same time.

3.4 IBM FileNet Capture

IBM FileNet provides applications for ingesting paper and image-based content for both structured and unstructured data with complete integration with the IBM FileNet P8 Platform. This tight integration gives businesses the ability to incorporate much of their document into the system with little human intervention. IBM FileNet Capture products automate control and classification during the capture process, which enhances IBM FileNet P8-based compliance by increasing accuracy and lowering the risk of lost or inaccessible information.

IBM FileNet Capture Professional is the main product that scans, indexes, and converts content to PDF and stores them in IBM FileNet P8. Capture ADR, Fax, and Remote Capture are add-on modules that provide form processing and recognition capability for IBM FileNet Capture Professional.

3.4.1 Capture process overview

Capture processes convert paper into digital documents that are a representation of the original paper. There are five steps in the typical capture process:

1. Create images using scan, Fax, or file import.
2. Process document, which involves image clean up and bar code/patch code recognition (Optional).
3. Acquire metadata, which is a manual step that requires operator typing.
4. Convert to PDF (Optional).
5. Record activator (Optional).
6. Commit to IBM FileNet P8 Content Engine.

Figure 3-6 on page 68 shows the basic capture functionality provided by IBM FileNet Capture. Incoming documents from the Fax add-on module can be processed before continuing the rest of the capturing process. The Scan module has document processing and image cleanup already incorporated, thus scanned documents go directly to indexing. The indexing function acquires metadata from the documents. OCR2PDF is PDF conversion, which is an

optional feature. Additionally, documents can be declared as records before the images are committed to the IBM FileNet P8 repository.

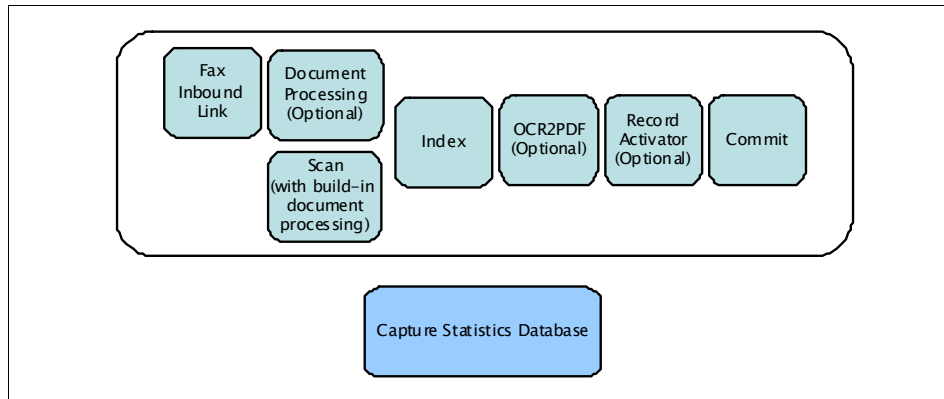


Figure 3-6 Basic capture functionality

Typically, paper-based content that requires capture is external to an organization, such as mailed correspondence, invoices, or technical information. This information serves to initiate, support, and further a business process. Much of the capture for paper-based documentation is centralized as an adjunct or extension to a mail room operation. Centralized capture or scan operations typically run in a specialized production environment. The need to quickly move information through such an operation requires multiple levels of expertise. There is also a high degree of validation and control to ensure that the paper-based information is moved correctly to a digital form.

Because of the steps needed to move information from paper to digital, capture supports a simplified queue system that allows batches of images to be automatically routed through the capture process. This simplified queue system is called a Capture Path, which we discuss in “Capture path” on page 72.

Using FileNet Capture Advanced Document Recognition (ADR) customers can meet the growing demands brought on by time-sensitive, mission-critical applications. Any enterprise with a high volume of documents of diverse types and extensive document-extraction needs will benefit from Capture ADR.

In the Advance Capture processes, Figure 3-7 on page 69, the Indexing step in Figure 3-6 is replaced by the following functions:

- ▶ Classification and separation
- ▶ DocReview (Document review)
- ▶ Recognition
- ▶ Correction
- ▶ Completion

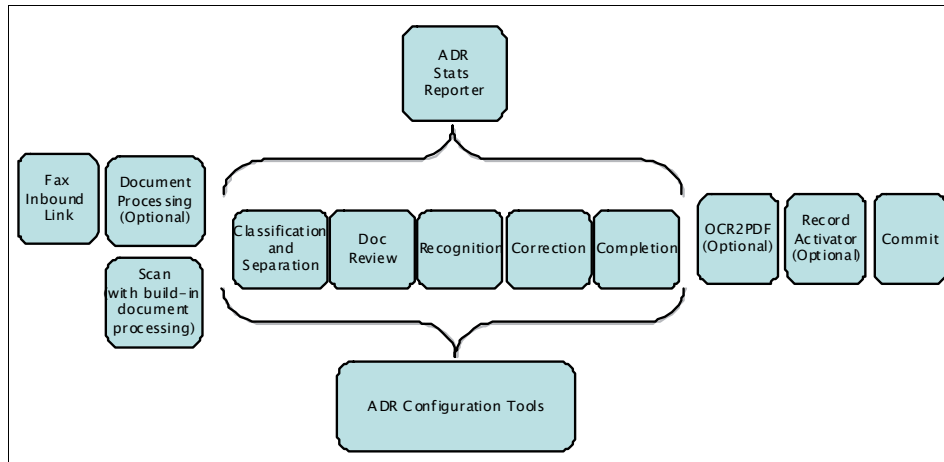


Figure 3-7 Advanced capture process

This is the most common and preferred process for most companies. We discuss these features in detail in later sections of this chapter.

These capabilities constitute the essence of capture that is directly supported by IBM FileNet Capture technology and provide the ability to tailor a capture solution to meet the changing needs and specific requirements of your enterprise. All document capture components, which includes assembly, document entry, document processing, file import and Optical Character Recognition (OCR) can be easily included or removed from the application.

3.4.2 Capture systems architecture

Figure 3-8 on page 70 shows the FileNet Capture in a distributed architecture and the capture system elements. We recommend adopting a distributed architecture for high-volume applications. Each of the steps in the capture process can be performed on separate systems.

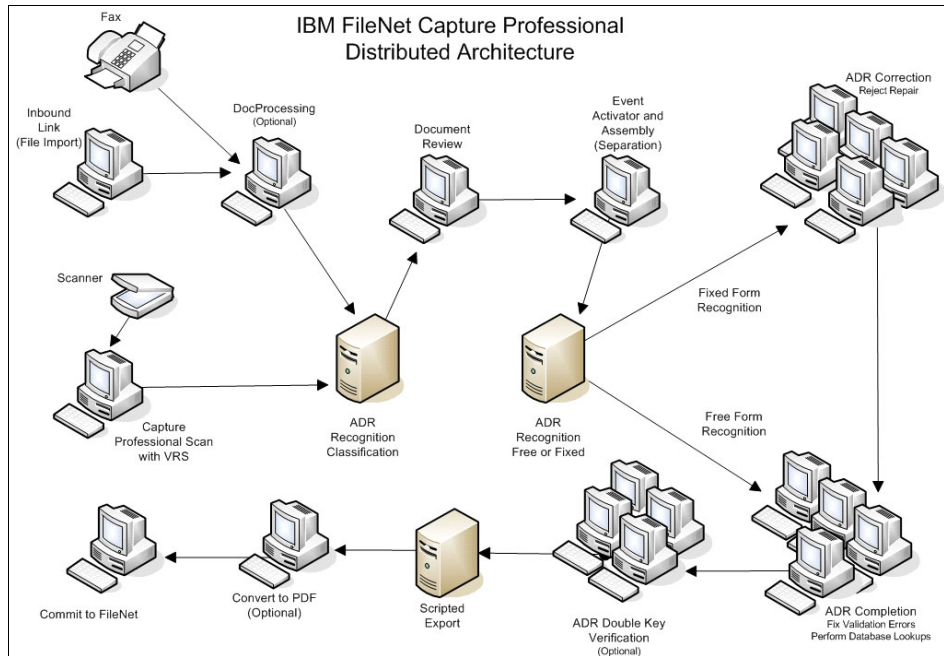


Figure 3-8 IBM FileNet Capture in a distributed architecture

You can have individual systems to perform file import, fax, and optionally document processing before the capture process goes through Advance Document Recognition (ADR).

3.4.3 IBM FileNet Capture products overview

Capture and capture technology are an extremely critical part of any successful ECM project. Basic capture functionality is supported by IBM FileNet Capture Professional and Capture Desktop. IBM FileNet Capture Professional also includes capture paths for automation, OCR and Patch/Bar code recognition, and manual indexing:

- ▶ Basic capture
- ▶ Capture paths to automate process
- ▶ Zonal OCR
- ▶ Patch/Bar code recognition
- ▶ Manual index
- ▶ Convert to PDF
- ▶ Set Record declaration property
- ▶ Commit to FileNet CE

FileNet Capture Desktop performs scanning and metadata extraction only and is a simplified capture solution. This subset of Capture Professional is non-distributed and contains no fax or OCR support.

The FileNet capture modules for Desktop and Professional support the entire range of batch capture functions and a collection of drivers for production level scanners and the major driver standards, including:

- ▶ ISIS
- ▶ Twain
- ▶ Kofax

Most applications have more advanced capture requirements. IBM FileNet offers the following product add-ons to Capture Professional:

- ▶ Capture ADR (Advanced Document Recognition)
- ▶ Fax
- ▶ Remote Capture

Capture ADR supports the following tasks:

- ▶ Data extraction/validation
- ▶ Automated Classification reduces manual document separator sheet insertion
- ▶ Separation
- ▶ Advanced OCR: Free and Fixed Form recognition
- ▶ ICR for constrained handprint recognition
- ▶ Mark-sense for bubble or check box recognition
- ▶ Bar code recognition
- ▶ Database validations
- ▶ Table processing
- ▶ Export to text, XML or CSV file

IBM FileNet Fax is an enterprise-class solution that provides integrated fax document management across the enterprise.

Remote Capture is a Web-based support for Internet Information Service (IIS) document import, index, and verification solution for remote workers. Enabling remote sites to securely perform tasks over the Internet helps to lower the costs of dedicated WAN connections and increases network availability.

Capture Professional and Capture ADR are the two key products for Capture support; therefore, in the next two sections we detail how these products expand IBM FileNet functionality.

3.4.4 IBM FileNet Capture Professional

IBM FileNet Capture Professional provides enterprise-level production. Capture can be automated in IBM FileNet Capture Professional in batches and it can also be scripted.

IBM FileNet Capture Professional functions

IBM FileNet Capture Professional provides its functions through capture path, batch template, setting collection, and Capture Toolkit.

Capture path

Capture path is a key concept in IBM FileNet Capture Professional. The Capture Path defines an automated sequence of document ingestion operations for which the batch is to be processed. The ability to configure and manage Capture Paths supports flexibility and efficiency in the construction of a production scanning environment. Because all paper-based document collections differ, the ability to combine capture functions should also differ, for example, it is common to receive imaged information from partners in excellent condition. In this case, some capture steps, such as image verification, can be eliminated from the capture path.

Batch template

A *batch template* determines what is done, and where it goes. It is created by selecting a Settings Collection and a Capture Path.

Settings collection

A *settings collection* holds configuration information that defines how Capture components are to behave when they are called to process a batch. In addition, the Settings Collection specifies the FileNet Repository Document Class. The Capture Path defines an automated sequence of document ingestion operations to process the batch, which we further discuss in the next section.

Batch scanning allows documents to be scanned and processed in groups, which allows for optimal system performance and operator efficiency.

Capture Toolkit

The Capture Toolkit is a part of Capture Professional and Capture Desktop that provides a rich set of sample applications, documentation, and other files that are necessary for developing Capture custom applications using the Capture components, for example, sample applications are provided to present a different user interface for scanning or indexing.

The toolkit is used to quickly build and customize document-capture tools to provide essential functionality. This quick build includes capture options that can

be customized, which includes document assembly, repository administration, local and multi-station automation, conversion from other systems, continuous scanning, error management, and custom components.

The toolkit takes advantage of the Capture architecture to automate document entry through built-in tools, such as Capture Paths and through custom implementations. The underlying COM objects give complete control over the user interface and Capture operations to manipulate repository servers and repository objects.

IBM FileNet Capture Professional components

IBM FileNet Capture Professional components support document ingestion operations that can be invoked while capturing documents. These components include Image Verify, Document Processing, Blank Page Detection, Patch/Bar code recognition, Event Activator, Assembly, OCR, Index, Index Verify, Merge, OCR2PDF, and Records Activator.

Image Verify

Image verification is used to display captured images for visual evaluation of image quality and page organization. Image verification is normally done before assembly but can also occur before assembled documents are committed. You can view the pages and manually reject pages, such as blank pages and separator sheets, and mark other pages for later rescan. You can also use image verification to review each of the pages in a batch.

Document Processing (DocProcessing)

DocProcessing provides a set of components to automate indexing and to improve image quality after scanning, faxing, or importing, which includes image clean up and Bar code/Patch Code recognition and is optional.

Blank Page Detection

When scanning duplex pages, you might acquire blank pages in your batches. While these blank pages truly represent the actual pages scanned, they take up space and waste paper if printed. With this component, you can detect and remove blank pages in your batches.

Patch/Bar Code Recognition

Patch codes are commonly used to separate batches, and bar codes separate documents. The bar code value can exist on a separator page that persists with the document during auto-indexing, even if the separator is deleted during assembly. Capture can interface with scanners and scanner drivers that can perform patch code recognition.

Event Activator

The Event Activator component can perform actions based on rules that you formulate. The Event Activator can perform three types of actions:

- ▶ Separating objects into folders, batches, and documents.
- ▶ Changing the name that is assigned to a folder, batch, and document.
- ▶ Switching to a different settings collection: supported when Event Activator is used with a batch separator rule.

Assembly

Document assembly is the process of sorting, organizing, and grouping individual pages into documents for subsequent indexing and committal. The Index component, for instance, cannot process documents that are not assembled. Assembly is commonly done on the batch level. A batch is usually assembled only once and only in one way: manually, ad hoc, or using a capture path.

Optical Character Recognition (OCR)

Optical Character Recognition, or OCR, converts parts of or all of scanned pages of machine print into editable text. Converting parts of scanned pages is known as Zonal OCR. The values that are found can become metadata or attributes of the document. The metadata can be used to route or index automatically.

Index

Indexing with IBM FileNet Capture is a coordinated process that uses index fields from the IBM FileNet server and settings and index fields from Capture, which includes metadata.

Indexing is typically done as late in the document capture process as possible to ensure that the maximum possible attributes are available for indexing. Indexing as late as possible in the document entry process is appropriate when manually typing index entries or when using the Capture auto-indexing feature.

Index Verify

Index Verify is a way to double-check selected index entries before a document is committed for Image Services only. The fields that are used for Index Verify are set up on the Image Services server at the same time that indexing for the document class is set up. Normally, Index Verify is used any time after a document is indexed or auto-indexed, but before it is committed.

Merge

Merge component combines multiple individual image files of the same or compatible type into a single multi-page file. Merge is used in the Content Engine environment because each document can contain only one file. The Merge

component combines multiple files into a single file to satisfy this requirement for Content Engine. The Merge component combines multiple JPEG/color TIFF, TIFF, or PDF images into a single file prior to committal. It can also combine multiple PDF images into a single PDF file or multiple TIFF images into a single TIFF file. Merge converts each JPEG/color TIFF image to a TIFF image and then merges the images into a single file that has a TIFF format.

OCR2PDF

The OCR2PDF module performs OCR on images and generates a PDF file with embedded text, which allows Full Text Search to be performed through IBM FileNet Content Manager's search engine.

Records Activator

Records Activator provides the capability to automatically assign records management related information of a document based on a default value for the document class, for the batch or for the document, for example, documents can be associated with a specific file plan based on their attributes, such as barcode value or state.

3.4.5 IBM FileNet Capture Advanced Document Recognition (ADR)

IBM FileNet Capture Advanced Document Recognition is a powerful suite of IBM FileNet Capture modules that provide advanced recognition and extraction capabilities. FileNet Capture Advanced Document Recognition automatically extracts hand-written and printed data from scanned document images, removing the need for costly and time-consuming manual keying. FileNet Capture Advanced Document Recognition can use the extracted data to automatically classify, separate, and route the document to a workflow queue to index the document for storage or to initiate a business process or transaction.

Figure 3-9 on page 76 illustrates the ADR capture process.

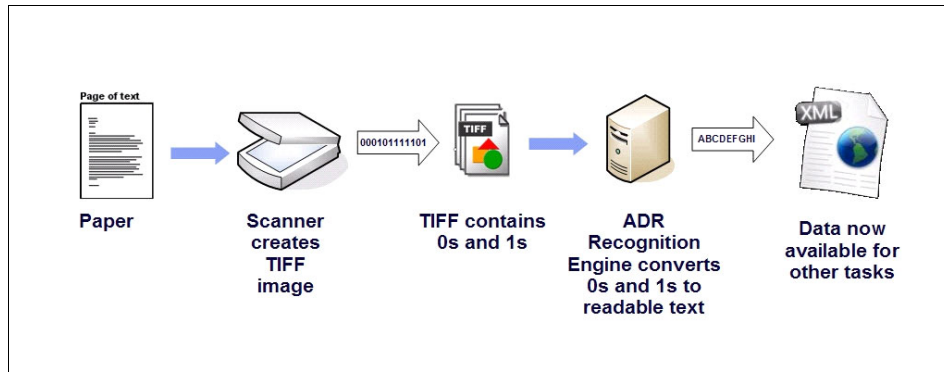


Figure 3-9 ADR capture process

ADR provides OCR, ICR, and marking (check box) recognition, both zonal (OCR to pickup values from a pre-specified area or zone, typically on a printed form) and full page. Capture ADR is an extension of the FileNet Capture Professional add-on. The product's advanced document recognition functionality includes:

- ▶ Advanced OCR
- ▶ Intelligent Character Recognition (ICR) for constrained, handwritten information
- ▶ Optical Mark Recognition (OMR) - (also mark-sense) for check marks, bubbles, and so on
- ▶ Free and Fixed Form recognition and processing
- ▶ Data extraction/validation
- ▶ Automated Classification
- ▶ Separation
- ▶ Database validations
- ▶ Table processing
- ▶ Export to text, XML or CSV file

Advanced functionality of ADR

The addition of the advanced capabilities to the FileNet Capture path provides a powerful set of recognition functions that result in substantial savings, efficiency, and reliability for scanning needs.

Advanced OCR/ICR

Capture ADR supports a recognition trainer that allows new machine print typefaces and handprint character sets to be added to recognition. The

recognition trainer provides a user interface to train the software to recognize new or difficult characters. Because of this feature, the system gets smarter and recognition results improve over time.

Mark-Sense (OMR)

Optical Mark Recognition processes arrays of check marks, bubble marks, and other non-text marks commonly used on forms.

Automated classification

Not to be confused with the functionality of the Classification products, this feature reduces manual document separator sheet insertion by classifying the documents correctly through recognition.

Forms recognition and processing

For semi-structured and unstructured documents that need correspondence, where the location of data items varies, IBM FileNet Capture ADR uses free-form algorithms to automatically locate the data items using knowledge of format, proximity to tags, and validation against business rules.

IBM FileNet Capture ADR uses a template approach for extracting data from fixed forms where the location of data items is known, such as loan application forms, surveys, remittances, and other forms. For semi-structured and unstructured documents, for example correspondence, where the location of data items varies.

The steps of forms capture are:

1. Document type recognition.
2. Document correction: To correct for scanning artifacts, such as stretch, skew, and shift.
3. Image cleanup: To remove noise, background, and others.
4. Segmentation: To locate the individual characters on each document image.
5. Recognition: To convert a character bitmap to a Unicode character code.
6. Application of a template: For structured documents or
Free-form Technology: To automatically locate data on semi-structured and unstructured documents.
7. Separation: The ability to determine document separation without the need for separator sheets.

3.4.6 IBM FileNet Remote Capture

IBM FileNet Remote Capture Services is both an application and XML Web Service. Its multi-tiered architecture allows the application, services, and data tiers to exist on the same box or on different boxes for optimum scalability.

Built on Microsoft's .NET technology, Remote Capture Services is a standards-based (SOAP, WSDL) mechanism that establishes a message-based protocol, which enables a client to discover and consume services that are offered by a server. Remote Capture Services utilizes the HTTP or HTTPS transport mechanism.

IBM FileNet Remote Capture Services enables geographically dispersed users to acquire, index, and verify Capture documents over the Internet or intranet.

3.4.7 IBM FileNet Fax/MFP

IBM FileNet Fax offers the Captaris RightFax Enterprise Server software for the receipt of inbound faxes. The RightFax Enterprise Server runs as a service so that the fax server can automatically logon during restart. Capture and Print utilize a component called FileNet Connector to integrate seamlessly to the RightFax Server. You can connect as many Capture and Print stations to the RightFax Server as needed. Used with Capture Inbound Link, faxed images can go directly into the Content Engine.

Filters might be configured to have a Capture Inbound Link station that processes only those faxes with specified attributes values, which gives control to the administrator to specify which faxes are processed by Capture Station(s), for example, Capture stations can process faxes based on user ID or billing code. Include/Exclude capability allows for configuring criteria where the attributes or eFaxes take advantage of the full suite of Capture components for processing, such as barcode detection, automated routing to selected document classes, and indexing based on fax attributes and barcode values, document assembly, image enhancement for cleaning up faxes, and commit.qual or not equal to the values specified.

Fax and Multi Function Printer/Device (MFP) support is an important variation of the several capture methods that are included in the FileNet capture add-on. The FileNet Architecture supports Right Fax servers.

Inbound faxing with IBM FileNet Capture integration

Incoming faxes are brought into your system through the IBM FileNet Capture Professional Inbound Link. Documents can be automatically routed and processed by any of the fax attributes, such as the phone number from which the

fax came, which helps to streamline processes that are based on your line of business.

Desktop Outbound Faxing

IBM FileNet Fax enables users to directly and quickly fax out content that is stored in Content Manager from their desktop. IBM FileNet FAX also allows incoming faxes to pass metadata directly to automated fax entries in to the repository.

3.4.8 Integration points

IBM FileNet Capture Professional is built using Microsoft's OLE Automation technology, which provides an object-oriented component-based architecture. This architecture allows third party components to interact seamlessly with Capture. Capture's modular architecture allows you to tailor a capture solution to meet specific enterprise needs. IBM FileNet Capture is completely integrated with all FileNet Repositories throughout the capture process, including:

- ▶ IBM FileNet P8
- ▶ IBM FileNet Image Services
- ▶ IBM FileNet Content Services
- ▶ IBM FileNet Records Manager

When connected to an IBM FileNet repository, IBM FileNet Capture uses the authentication method that is consistent with that FileNet repository. FileNet Capture performs real time lookup of document class and field definitions that are configured in the FileNet repository. For committal, FileNet Capture uses the FileNet repository's APIs to store documents and metadata appropriately without customizing the software.

In addition, there are components that allow the exposure of the Records Manager File Plans for declaration and retention, which allows documents to be declared as records in the capture path, which means that capture works with Records Manager giving an organization the ability to bring scanned images under records control immediately.

VBScript functions can be used to manipulate the data that FileNet Capture recognizes. These functions are also completely integrated with all FileNet Repositories throughout the Capture process.

3.4.9 Capture summary

Capture encompasses a wide range of functionality. This range of functionality is driven by the range of information representation combined with the technologies

developed over the years that cope with this variety. The IBM FileNet Capture expansion product set focuses on providing a management framework to allow customers to realize the appropriate application of these technologies to improve business operations. In particular, these products allow for efficient implementation of a production capture environment to quickly capture, organize, control, and utilize their documents.

3.5 Summary

The content ingestion expansion products provide core applications to quickly, efficiently, and intelligently ingest documents into the IBM FileNet P8 repository. These products not only add this content, but they can annotate and organize the information to make it more useful to the customer. By adding key metadata, indexing, and declaring records, IBM FileNet Capture and the IBM Content Collector family products expand the IBM FileNet P8 Platform by faxing, scanning, and importing files in critical business applications. These products make automation and integration simple and powerful simultaneously. Businesses gain greater knowledge and control over their mission critical information while increasing their agility in responding to market changes.



Expansion products for connectors and federation

In this chapter, we discuss connectors and federation products that enable content resources to be centrally available and usable, which enables businesses to maximize these resources for IBM FileNet P8 systems.

The topics that we cover are:

- ▶ 4.1, “Connectors and federation products overview” on page 82
- ▶ 4.2, “IBM FileNet Application Connector for SAP R/3” on page 82
- ▶ 4.3, “IBM Content Integrator” on page 88
- ▶ 4.4, “Content Management Interoperability Services” on page 91
- ▶ 4.5, “IBM FileNet Services for Lotus Quickr” on page 93
- ▶ 4.6, “IBM FileNet Connectors for Microsoft SharePoint” on page 100
- ▶ 4.7, “Summary” on page 106

For an overview of all of the IBM FileNet P8 expansion products, refer to 3.1, “Expansion product overview” on page 56.

4.1 Connectors and federation products overview

Expansion products for the IBM FileNet P8 Platform protect and expand corporate investments in heritage file repositories. Businesses can maximize these resources by using them in new and intelligent ways while they remain in the heritage systems. A key differentiator for IBM FileNet P8 is that documents from databases and team collaboration tools can be made centrally available and usable assets through connectors and federation products, which include:

- ▶ IBM FileNet Application Connector for SAP R/3
- ▶ IBM Content Integrator
- ▶ Content Management Interoperability Services
- ▶ IBM FileNet Services for Lotus Quickr
- ▶ IBM FileNet Connectors for Microsoft SharePoint

Teamwork collaboration products, such as Microsoft SharePoint and Lotus Quickr have repositories. Connectors and services by IBM activate this content with business processes. Content for Microsoft SharePoint and Lotus Quickr can also be imported, copied, or moved and linked through these products. IBM also embraces the CMIS architecture, so that products by Microsoft, SAP, and other vendors can be incorporated using one simplified, modular, industry-standard method.

Businesses break down walls of communication by ensuring that this connectivity between repositories. They can mine new information from previously inaccessible data and can control documentation to meet compliance requirements. Information becomes an active, integral part of business processes. IBM FileNet P8 expansion products help businesses make better decisions, faster.

4.2 IBM FileNet Application Connector for SAP R/3

IBM FileNet Application Connector for SAP R/3 enhances SAP business infrastructure with archival and portability support through the content management and business process management of IBM FileNet P8 Platform. These flexible application connectors improve productivity by providing document enabling for SAP and mySAP ERP. They support compliance requirements with corporate records retention management and enhance controls by enabling SAP workflow.

IBM FileNet P8 provides two connectors for SAP:

- ▶ IBM FileNet Application Connector for SAP R/3 (ACSAP R/3) provides critical archiving capabilities for the SAP system
- ▶ IBM FileNet Application Connector for SAP R/3 Enterprise™ Portal (ACSAP EP) integrates the IBM FileNet repository seamlessly with the SAP Portal

4.2.1 IBM FileNet Application Connector for SAP (ACSAP) R/3

The main function of Application Connector for SAP R/3 is to store, link, and retrieve data and documents to and from IBM FileNet repositories in SAP R/3 applications. Using IBM Application Connectors for SAP, businesses save money by archiving documents out of the SAP system and reducing costly paper. They improve business processes and productivity by using Archive-Link enabled modules. ACSAP R/3 integrates SAP transactions to documents that originated outside of SAP, such as vendor invoices, quotes, faxes, CAD drawings, or any other type of electronic documents, through capture, display, storage, and management of these documents. After they are linked to SAP transactions, they are available to SAP users through the SAPGUI and optionally available to other interested parties outside of SAP using IBM FileNet front-end applications.

The document linking and viewing capabilities of Application Connector for SAP R/3 enables SAP users to easily find, manage, and link documents and folders to SAP transactions. Also, leveraging IBM FileNet image handling capabilities, large volumes of documents can be captured, ingested, and made available for linking in both manual and automated modes.

4.2.2 Data and document archiving

ACSAP R/3 supports all three inbound archiving scenarios that are defined by SAP, early, late, and simultaneous linking, and barcode processing for fully-automated linking. Early, late, and simultaneous linking are all inbound archiving. These documents are processed while coming into the system. Outbound archiving links documents that are created by the SAP system, such as lists or invoices.

In *Early Archiving*, a document is captured into a FileNet repository and made available for linking before the associated SAP business object is created. This scenario is useful in high volume document entry operations, where separation between document capture and SAP business objects processing is key, for example, a highly automated and distributed scanning process can be implemented in the mail room using FileNet Capture independently from the

business process in SAP. It is also the first stage of any incoming document scenario where the document is processed using SAP Business Workflow.

In *Late Archiving*, the creation and processing of the SAP business object comes first and linking to the corresponding supporting document happens later in the process. In practical terms, this process is more in line with the traditional paper-based process.

In *Simultaneous Archiving*, all document entry and SAP object processing steps are carried out by the same SAP user. Overall the process is the same as in Early Archiving except that the SAP work object, which is created at link time, is assigned to the current user.

In barcode processing, a barcode value is generated to uniquely identify the paper document and SAP transaction that the document needs to be reconciled with. At scanning time, the barcode value is then automatically read off the paper document to be used by a background task to link the document to the target transaction.

The IBM FileNet Connector for SAP improves document availability while reducing the cost of document archiving. Administrative costs are also lowered in the process and tracking of the status of documents and approvals occurs directly in the SAP Business Workflow/ Webflow. The administration and configuration of the connector is Web based, and the client is zero footprint, which means that it requires no download. It is SAP certified for many SAP interfaces, including BC-HCS, BC-AL, and AL-LOAD.

4.2.3 Application Connector for SAP Enterprise Portal (ACSAP EP)

ACSAP EP provides several important features to SAP clients. Documents and metadata in FileNet Content Manager are seamlessly available and tightly integrated in the SAP Portal search engine (Text Retrieval and Information Extraction - TREX). This integration means that documents that are stored in FileNet can be viewed in the portal, and search results from FileNet are also accessible in SAP. Collaboration is supported with comments, feedback, ratings, subscription, and alerts through integration with the SAP Knowledge Management (KM). The SAP Administrator can use the rich functionality of the FileNet workplace right within their portal, using a portlet with a URL iView. The administrator can configure it through the SAP Portal administration, so the interface is intuitive.

4.2.4 IBM SAP connectors architecture

The IBM SAP support helps SAP customers make better decisions by having the right information, at the right time. With ACSAP R/3, IBM document-enables the mySAP application, integrates with mainstream business applications, and has powerful document entry. The intuitive interface and modular architecture make it flexible, and easy-to-use. Figure 4-1 shows a high level of the IBM SAP connectors architecture.

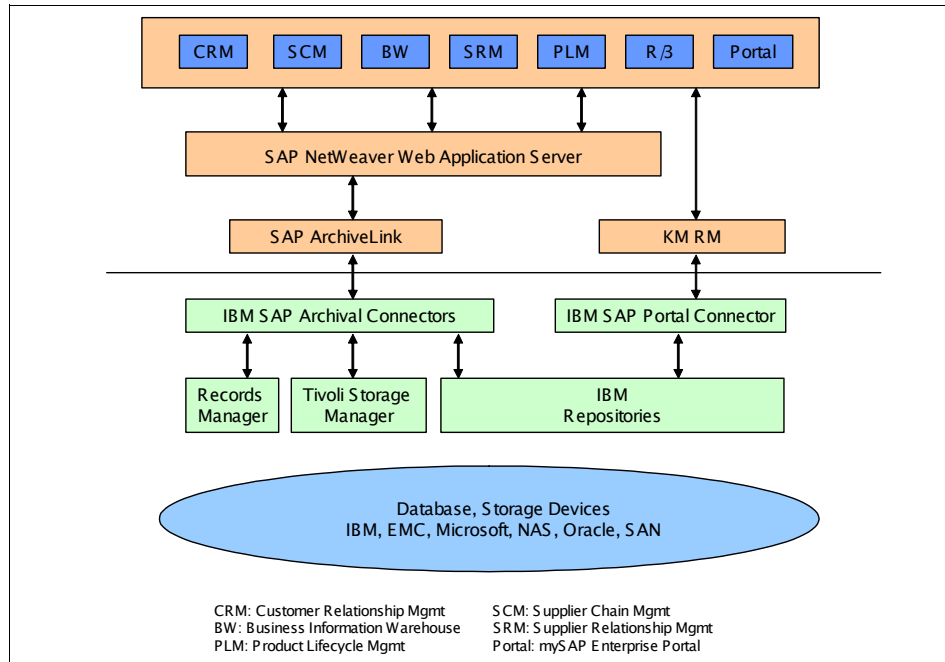


Figure 4-1 High level IBM FileNet P8-SAP architecture

In Figure 4-1, the orange section contains all of the SAP products and applications that they provide. The SAP Web Application Server, part of the NetWeaver platform, interfaces with the SAP ArchiveLink and the Knowledge Management Repository Manager (KM RM) interface, which is part of the SAP portal. The green section is the portion that IBM provides, which includes the ACSAP R/3 interface with the ArchiveLink for archival. ACSAP EP connects with the KM RM to provide seamless interoperability with the IBM FileNet Content manager repository.

4.2.5 ACSAP R/3-J2EE architecture

ACSAP R/3-J2EE 2.1 connects SAP with IBM FileNet through the ArchiveLink interface. ACSAP R/3 is operating system independent and can run on a variety of application servers. On the server side, MySAP ERP interfaces through the application server to the SAP Java Connector. The ACSAP R/3 connector then communicates both to either, or both of, IBM FileNet Image Services and the IBM FileNet Content Engine, as is seen in Figure 4-2.

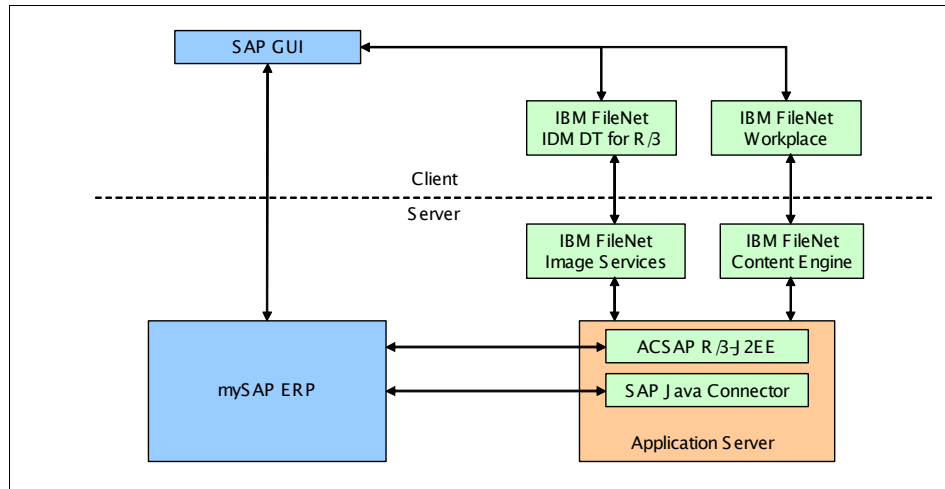


Figure 4-2 ACSAP R/3 J2EE architecture

In Figure 4-2, on the client side, the SAP GUI connects to the Image Services user interface IBM FileNet IDM DT for R/3 and the IBM FileNet Workplace user interface as well. In this way, both IBM FileNet Image Services and IBM FileNet P8 are supported.

4.2.6 ACSAP EP architecture

ACSAP Enterprise Portal (ACSAP EP) runs within the J2EE environment on the SAP NetWeaver® Web Application Server, which connects the SAP portal to IBM FileNet repositories. ACSAP EP is supported on all Windows and UNIX® platforms that the SAP Enterprise Portal supports. Figure 4-3 on page 87 shows the architecture.

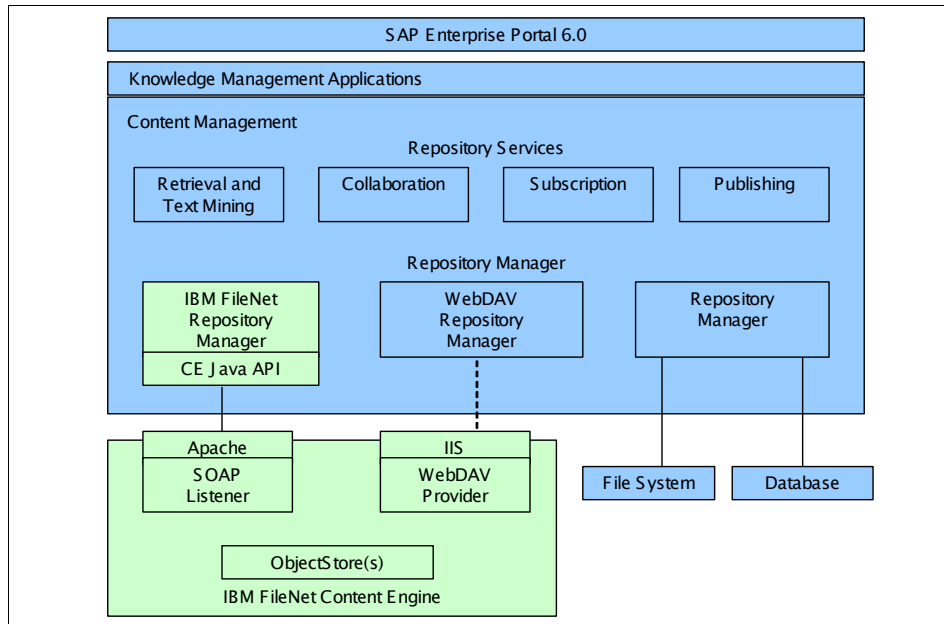


Figure 4-3 ACSAP Enterprise Portal: Knowledge Management architecture

In Figure 4-3, The blue portions of the architecture are provided by SAP as part of their Knowledge Management and Content Management. The green portions are the integration with IBM FileNet through the Content Engine Java API directly into the IBM FileNet Content Engine.

Users access information through the portal, which in turn accesses the information through the Repository Manager. The Repository Manager connects through a Repository Framework, which uses the Java APIs to communicate with the Content Engine.

Knowledge Management is built on top of the Repository Framework and provides the entry point to finding unstructured data in the SAP environment. Knowledge Management is also used to navigate and search the IBM FileNet Content Engine repository, which is also where collaboration activities, such as subscriptions, occur. Metadata is also fed through this interface.

4.2.7 SAP summary

With ACSAP R/3, IBM document-enables the mySAP application, integrates with mainstream business applications, and features powerful document entry. ACSAP EP's intuitive interface and modular architecture make it flexible and easy-to-use IBM FileNet repositories and functionality in the SAP environment.

Using these products, SAP customers can take advantage of the strengths of the IBM FileNet P8 Platform.

4.3 IBM Content Integrator

IBM Content Integrator (formerly known as IBM WebSphere Information Integrator Content Edition) is an important tool that helps businesses reuse and integrate existing content in Enterprise Content Management. Using Content Integrator organizations can reduce risk and costs by avoiding forced data migrations and associated application rewrites and redeployments. Simultaneously, they achieve a high degree of control over disparate content repositories. A key example is the ability to deploy an enterprise level records management solution while avoiding moving existing data in other content repositories and associated applications. Using Content Integrator in combination with IBM FileNet P8 organizations can realize the benefits of IBM FileNet P8 functionality immediately without migrating content, which is possible because Content Integrator technology provides a clear connection model to IBM FileNet P8 and the additional benefits of IBM FileNet P8 content management and business process management features.

Content Integrator enables speed and business agility driven by the dynamic nature of organizational and technological changes. Some other reasons to use Content Integrator are:

- ▶ Mergers and acquisition activity
- ▶ Compliance initiatives
- ▶ Cross-silo business process improvement projects

Content Integrator technology provides a fast and effective solution to these issues. The features of Content Integrator that allow these benefits include content-focused APIs with tight IBM FileNet P8 integration points, scalability supported by the robust IBM FileNet P8 architecture, and proven deployment options and methods.

4.3.1 Content Integrator architecture

Content Integrator has three main layers:

- ▶ Developer and User Services: Includes the API interfaces and URL and HTTP access that are used to interact with custom and enterprise applications

- Federation Services: Used to federate content and tools to assist with federation
- Integration Services: Used to access the content in other repositories

In Figure 4-4, we outline these layers.

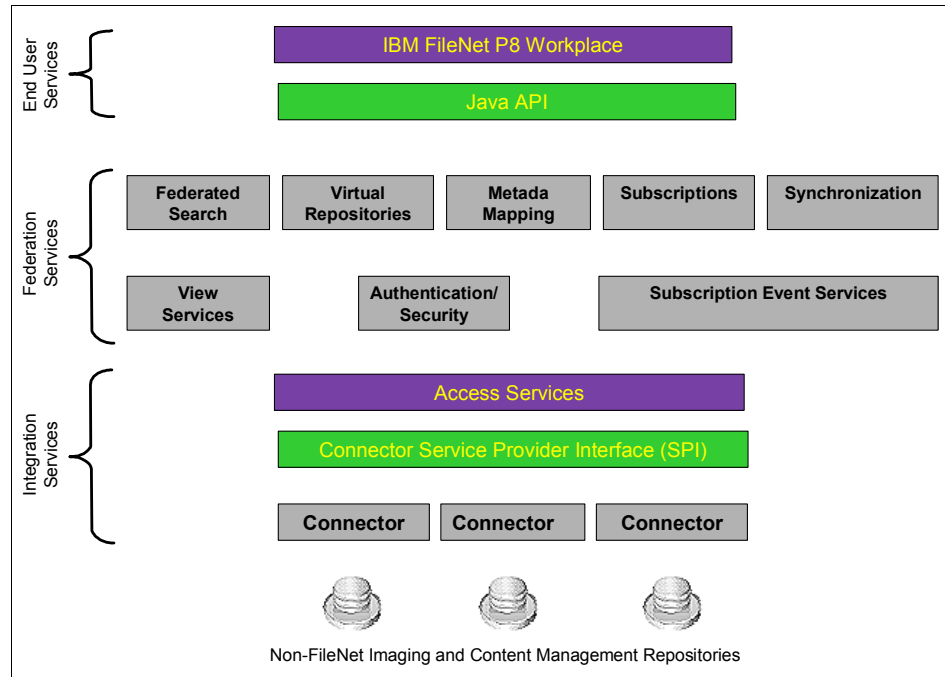


Figure 4-4 Content Integration architecture

Developer and User Services

Using the Integration API developers can write content management and workflow applications that span multiple systems while treating them generically. This ability is similar to the CMIS approach that we discussed in 4.4, “Content Management Interoperability Services” on page 91. This object-oriented approach hides the dissimilarities in repositories from the developer.

The HTTP access is a faster way to address content, reducing possible latency for distant repositories. All addressable content (such as folders, content items, work items, and queues) have a Universal Resource Name (URN) as a unique identifier. This URN can be used to construct a URL to retrieve any item through a standard HTTP request to the Content Integrator server.

The Web Services API uses a SOAP interface and includes a WSDL file that defines the API; therefore, the developer is free to use a variety of languages to retrieve documents.

Federation Services

Federated content query is used to find documents. These search options include full-text queries and index creation.

The data map designer maps information in one repository to the same kind of content in another, such as Last Name and Family Name.

View services displays documents and images in a Web browser format. Developers are relieved of the requirement to provide specific content viewers. Images can also be manipulated and annotated.

Authentication and security enables single sign-on functionality that streamlines retrieving data from multiple repositories and authentication requirements.

Integration Services

Session pooling allows reuse of connections to repositories, limiting the number of connections and authentications and cleaning up connections when unused.

The administration graphical tool is provided for configuring connectors, subscription event services, logging, and other Content Integrator settings.

Connectors interpret Content Integrator APIs for repository APIs. This is the encapsulation of the specific storage requirements that supports the object-oriented design of Content Integrator applications.

The Integration SPI is used to customize connectors and to develop new ones.

The RMI proxy uses Java Remote Method Invocation (RMI) to connect to repositories so that one Java Virtual Machine (JVM™) object can invoke methods on an object that runs in another JVM.

The Web Services proxy is deprecated and connects to repositories using SOAP, which is useful when programming in a language other than Java, such as a Microsoft .NET client.

4.3.2 Content Integrator summary

Although in this section we do not go into depth into all of the possible Content Integrator connectors and functionality, it is clear that the ability to use content in existing repositories is a key requirement for most businesses. Content Integrator has proven scalability and functionality that can meet the demands of

integrating processes, documents, and images throughout the extended enterprise. With Content Integrator, IBM FileNet delivers the right information at the right time, no matter where the content is stored.

4.4 Content Management Interoperability Services

Content Management Interoperability Services (CMIS) is an industry-wide specification released that supports a new area of openness in Enterprise Content Management. It provides Web Services and Web 2.0 bindings for integration options, which is a Web services standard for interacting with Enterprise Content Management systems. All participating companies (including Microsoft, EMC, Alfresco, OpenText, SAP, and Oracle) have prototypes, so this is a valid specification with proven implementations. Many applications will use this going forward with SAP NetWeaver being one of the prime examples going forward and MicroSoft SharePoint and Lotus Quickr.

Using this specification in applications unlocks the business value of content because it can be managed from wherever and however the designer chooses. The SAP and IBM FileNet ECM prototype of this implementation makes the integration between the two systems deeper with minimal customization and configuration.

Companies often have multiple content management systems. This is a way to seamlessly communicate with them in a standard way. By providing a services-oriented architecture for interacting with an ECM repository, ECM applications can use CMIS loosely-coupled to individual repositories, rather than more tightly integrated, which enables companies to use CMIS interfaces a la carte rather than having to invoke the full-set of CMIS interfaces and allow applications to be built in a Services Oriented Architecture.

CMIS does not replace existing interfaces; instead, it enables companies to do rapid application development with a least common denominator set of interfaces.

4.4.1 Integration and connection points

CMIS allows the following services for object types and repository information:

- ▶ Create
- ▶ Read
- ▶ Update
- ▶ Delete objects
- ▶ Filing in zero, one, or multiple folders

- ▶ Navigate the folder hierarchy
- ▶ Create and view document versions
- ▶ Search, including full text search.

Control of security must be handled by the repository.

The prototype for SAP and ECM shows the repository integrated as though it were a mapped drive. As Figure 4-5 shows, the repository acts seamlessly as part of the product. The user can access it intuitively using the same methods as the rest of the product.

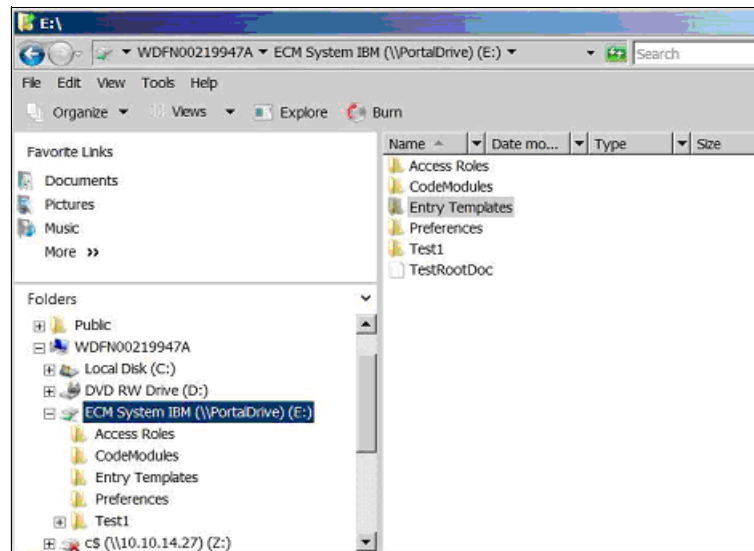


Figure 4-5 Prototype of the interface between the IBM FileNet repository and SAP

Using the CMIS protocol, the documents and their metadata are directly viewable from the SAP drive view. This application was created rapidly and required minimum product-specific configuration and coding to develop.

The full specification of CMIS is at the following Web site:

<http://www.ibm.com/software/data/content-management/cm-interoperability-services.html>

CMIS is not WebDav, but it is a new protocol. It supports Simple Object Access Protocol (SOAP) and Representational State Transfer (REST/Atom) based on Atom/APP.

The delivery vehicle (the ear file) contains the services handler. Search requests are handled exclusively by REST protocol, which is then mapped appropriately

for the correct search syntax; likewise, other requests that are handled by the REST and Web services handler are then mapped to the IBM FileNet data model. IBM FileNet uses Java APIs to communicate with the services, and IBM FileNet uses EJB as the transport layer.

4.4.2 CMIS summary

Although still being finalized, this specification is expected to be approved by the OASIS standards committee in 2009. IBM ECM is supporting CMIS with upcoming products and current announcements, which will expand the ability of users to use their existing systems in innovative ways with better interoperability between ECM products regardless of the vendor and helps to provide clients with more choice and lower costs when it comes to basic ECM needs. The CMIS standard also opens new possibilities for business partners and systems integrators to easily and effectively integrate Lotus and IBM ECM products.

4.5 IBM FileNet Services for Lotus Quickr

You can unlock your enterprise content with IBM FileNet P8 so that the content is more accessible across the organization from the tools that you use every day. With Lotus Quickr and IBM FileNet Services for Lotus Quickr, you combine collaborative authoring and sharing of everyday business content with the structure, business process management rules, and classification and discovery models that IBM FileNet P8 provide.

Using Lotus Quickr teams can quickly and easily work together. It consists of an overall Web site with various team places and a connector integration. The connector integration unites applications, such as Microsoft Outlook, Lotus Notes, and Windows desktop with Lotus Quickr places. These team places make it easy to share information. They are customizable and configurable to meet each team's requirements with items, such as calendars, announcements, to-do lists, blogs, RSS feeds, libraries, and many more.

The Quickr user interface with ECM products results in rich application integration. You use the Quickr Web user interface for collaboration, and you use IBM FileNet P8 to archive documents for compliance purposes and to drive business processes.

Adding IBM FileNet Services to the Quickr software gives workflows and business process management, centralized control of content and content types, and better scalability.

Lotus Connectors integrate directly with the desktop and various applications and allow users to interact with FileNet repositories as though they were any other folder. They can access the content no matter what environment they are in. Connectors allow for both import and access to content in FileNet repository. The IBM FileNet Services for Lotus Quickr provides the connectivity between Lotus Quickr and IBM FileNet repositories. Basic content services can now be performed against IBM FileNet repositories using Lotus Notes, Sametime, Microsoft Outlook, Lotus Symphony™, and Internet Explorer®. This integration is enabled by Lotus connectors. The integration is available for Quickr J2EE with IBM FileNet P8 v4.0.2 and Content Manager 8.4 FP1A. Certification for IBM FileNet P8 4.5 and Content Manager 8.4.1 is expected to be available for integration in 2009. Currently, support for Domino is not available.

A common scenario when using both products are: Documents are stored in the Quickr libraries. Then some documents, such as employee records, are needed by business for collaboration. In this case, you want to put these documents under control, such as in IBM FileNet P8 repository. You can do so with the help of IBM FileNet Services for Lotus Quickr. For some other documents, such as direction document to a luncheon place, these documents are not needed for any collaboration; therefore, they do not need to be archived or controlled within the IBM FileNet P8 repository.

Additionally, other IBM FileNet expansion products, such as IBM FileNet Records Management and e-Discovery extensions are available and help to preserve, protect, and facilitate reuse of Quickr documents. Together, the IBM FileNet Services for Lotus Quickr leverages the ease to use user interface and team collaboration of Lotus Quickr and the power of active content and business process management of ECM.

In the first release, the key functionality delivered includes Publication (move/copy) to and link to ECM, search, and the Document Library Viewer. Other integration points also expand the functionality exposed in the Quickr Web site.

4.5.1 Integration points

There are two main integration points. For the Quickr Web user interface, FileNet functionality is accessed through the Web pages using appropriate Web browsers. Quickr Connectors access FileNet using configured team places, as we describe in the following sections.

Quickr Web user interface

The Quickr Web user interface with IBM FileNet Services for Lotus Quickr connects to FileNet in three ways out of the box. One way to connect is through

the Publish command (move, copy, and link), search, and finally via the Library Viewer Portlet. The Publish command and search capabilities are available in Quickr libraries when installed and configured on the Quickr and FileNet servers. The Library Viewer Portlet is configured when adding them to Quickr Places, which are team Web sites.

A Library is many things in Quickr: the Quickr version of a repository area, the name of a page that views it and the portlet on that page that views it. There are two similarly named portlets: the Library Portlet, and the Library Viewer Portlet. A Library Portlet views Quickr stored documents, and a Library Viewer Portlet views FileNet or CM documents. The Library Viewer Portlet is a direct connection to ECM repositories with a limited level of interaction. The user can view content and metadata and navigate through the Library. It has less functionality than a Library Portlet, for instance, publish is not currently available from a Library Viewer Portlet.

To add the Library Viewer Portlet to a Quickr Place, the user must log in as a manager or higher level role. Using the customization widget or advanced customization, they can choose the Library Viewer Portlet. After the portlet is added to a page, the manager or administrator can then configure the portlet to use the appropriate repository location, as shown in Figure 4-6.

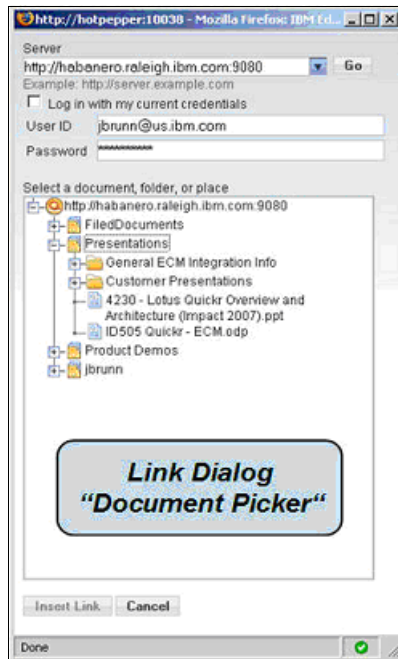


Figure 4-6 The Document Picker dialog box

The correct URL, port, and user ID and password combination are entered. Then the contents of the location are selected. The portlet can be set to use a particular folder as a starting point, which can be defaulted. When publishing from Quickr Places, it can be configured to prompt for metadata.

This same dialog is used to select locations to publish to and for links for blogs and wikis. In a Quickr Library, a user who has publishing rights can select the Publish action for a document. They can also choose the method of publication: move, copy, or link. Linking means that the content is moved into the FileNet repository, and the Quickr Library has a link directly to that document. All three of these selections allow for FileNet to take action on that content with workflows or records management. When publishing from Quickr Places, it can be configured to prompt for metadata.

In the Search user interface, the user can choose a scope to search over from a list that is configured when search is configured. This scope tells the Search tool what repository location to search over, such as a particular folder in a particular repository.

There is also the linking of existing content in IBM FileNet P8 and making it available in Lotus Quickr Web user interface. The Quickr Library viewer is showing a folder as a place (or a folder in a place) in Quickr Web user interface.

Lotus Quickr Connectors integration

Lotus Quickr connectors provide desktop integration (direct access) to IBM FileNet P8 and IBM Content Manager (CM8) content through Quickr Services. This integration gives users the ability to take advantage of IBM FileNet repository and business process capabilities regardless of the environment they are in: e-mail (Notes and Outlook), documents (Lotus Symphony and Microsoft Word), or Sametime instant messaging.

On the desktop, content can be dragged and dropped into Quickr Places to add them into the IBM FileNet repository. The user is prompted to publish the document or save as draft. Draft means that it is visible only to the owner who is editing the document. The connectors also can view, create, and restore versions of documents. Properties, that is the metadata of an item, can also be viewed, added, and modified. Figure 4-7 on page 97 shows the seamless integration of Lotus Quickr connectors.

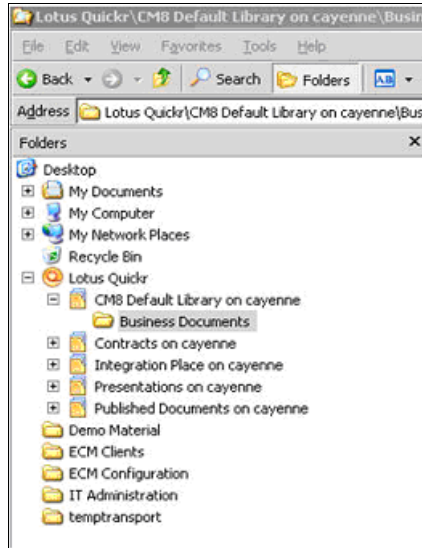


Figure 4-7 Lotus Quickr connectors integrate seamlessly

For Sametime, the same source selection dialog is available. Content can be browsed and selected in the same manner for linking.

Protocols

The library portal view in the Web UI uses HTTP calls to the Portal Server deployed in WebSphere Application Server, which in turn uses REST and Web Services to connect to the IBM FileNet Services for Lotus Quickr. The IBM FileNet Services for Lotus Quickr use EJB to communicate to FileNet, while IBM Content Manager uses JDBC, which occur using published IBM FileNet and IBM Content Manager APIs.

Desktop applications, such as Internet Explorer, Microsoft Office, Lotus Notes, Sametime, and Symphony, use Quickr Connectors, which in turn makes REST and Web Services calls to communicate with IBM Content Manager Services. See 4.5.2, “Architecture” on page 97. The REST services are ATOM based.

4.5.2 Architecture

The IBM FileNet Services for Lotus Quickr are delivered in two parts. The Services are deployed in WebSphere Application Server and can be deployed in the same Application Server where other IBM FileNet or IBM Content Manager applications are deployed. However, we recommend that IBM FileNet Services for Quickr be deployed in a separate instance of an Application Server. The

Lotus Quickr-ECM feature pack (which includes the Web portlets) must be installed on the Lotus Quickr server.

Figure 4-8 shows IBM FileNet Services and IBM Content Manager Services for Lotus Quickr, and IBM FileNet P8 and IBM Content Manager all are installed on WebSphere Application Server. The IBM FileNet Services and P8 4.0.x can be deployed in the same Application Server, although we recommend that they use separate instances. The IBM Content Manager back end does not need an application server. The Lotus Quickr 8.1 box on top of the Services for Lotus Quickr represents the Quickr Connectors.

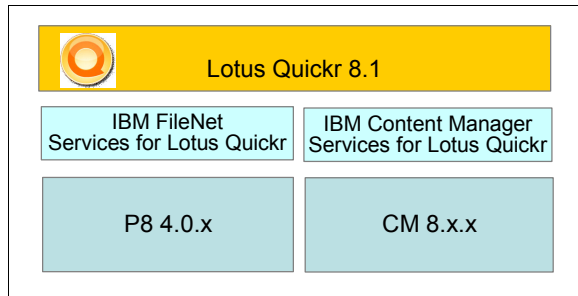


Figure 4-8 Lotus Quickr modules

Desktop applications, such as Internet Explorer, Microsoft Office, Lotus Notes, Sametime, and Symphony use Quickr Connectors, which in turn makes REST and Web Services calls to communicate with IBM ECM Content Manager Services, as shown in Figure 4-9 on page 99. All of these sources are presented to the applications in the same manner with the end goal being that interaction with data from any source is treated the same as any other.

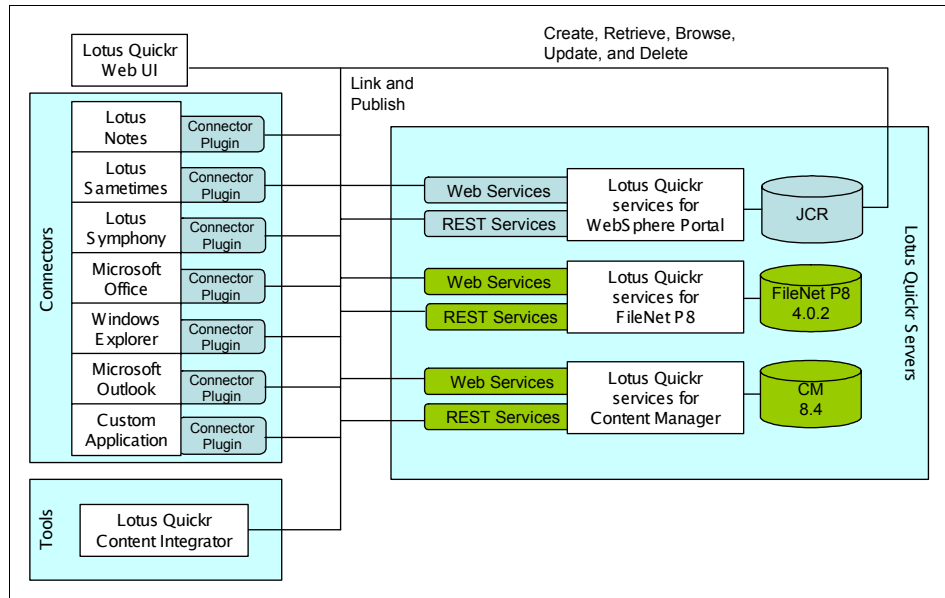


Figure 4-9 Services for Lotus Quickr connect using the Lotus Quickr connectors

The Quickr Web user interface communicates directly to the Lotus Java Content Repository (JCR) for most actions. Link, Search, and Publish actions use the Web and REST services.

Figure 4-10 on page 100 shows a more detailed architectural diagram where the delivery vehicle (the ear file) contains the services implementation. Search requests are handled exclusively by REST protocol, which is then mapped to the repositories search syntax; likewise, other requests are then mapped to the IBM FileNet P8 APIs. IBM FileNet P8 uses Java APIs to communicate with the services, using EJB as the transport layer.

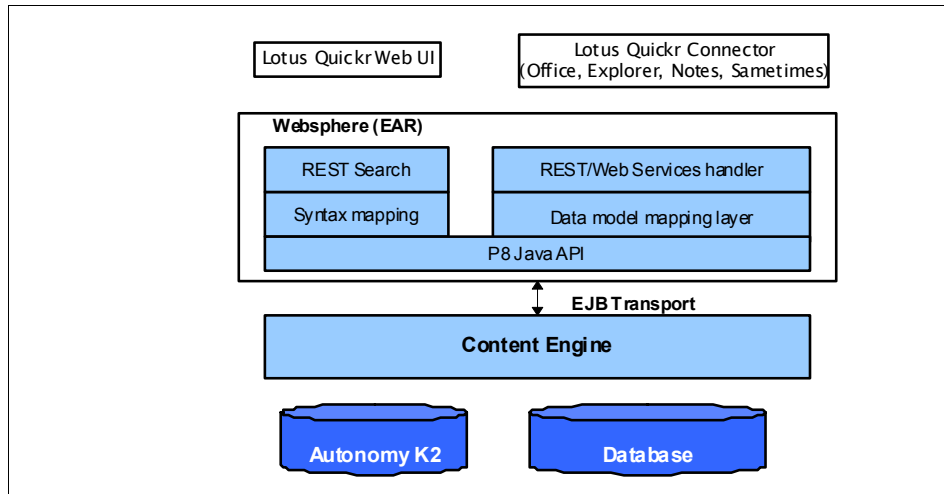


Figure 4-10 Quickr REST connections

4.5.3 Quickr summary

IBM FileNet Services for Lotus Quickr provide an intuitive collaboration environment with business process integration. Work can be accelerated and optimized, and corporate assets managed and protected. All of these features can be accomplished without requiring additional effort or training on the part of the user. Together, Lotus Quickr and the IBM FileNet P8 Platform create an environment where teams work together to accomplish more, faster.

4.6 IBM FileNet Connectors for Microsoft SharePoint

One of the core design points of any enterprise content management platform is to provide a wide variety of methods for storing and retrieving content. The IBM FileNet Connectors for Microsoft SharePoint is another example of an out-of-the-box integration to a widely used third party collaboration tool. Team collaboration is a key driving force in current business work practices, and the FileNet Connectors facilitate this ability.

The connectors are delivered as two integrated product offerings:

- ▶ IBM FileNet Connector for SharePoint Document Libraries
- ▶ IBM FileNet Connector for SharePoint Web Parts

The combination of these two offerings support the IBM FileNet P8 Platform's content management capabilities, so SharePoint users can view all content

across the entire enterprise. The connectors allow organizations who use SharePoint for its collaboration capabilities to gain additional capabilities, such as enterprise scalable records management and business process management, which can be delivered within the SharePoint interface. In this way, users can remain in an environment they are comfortable with (for example, Microsoft) while accessing all of the power and functionality of the IBM FileNet P8 Platform.

The Connector for SharePoint Document Libraries is the internal glue that makes the solution work, which gives high performance enterprise level integration. The Web Parts are the external facing integration that the user participates in, which provide a seamless look and feel to the user, so no new learning is required to take advantage of FileNet's strengths. In the next sections, we describe these product offerings in more detail.

4.6.1 IBM FileNet Connector for SharePoint Document Libraries

The Connector for SharePoint Document Libraries is a rules-based engine that monitors a chosen set of document libraries within a collection of SharePoint sites. Based on those rules, the documents are either moved or copied into the IBM FileNet content repository. When teams work together on documents or other information, it can be gathered, managed, controlled, and activated as content in FileNet repositories, which safeguards infrastructure investments by seamlessly integrating Microsoft SharePoint products and FileNet ECM solutions.

Typically, Microsoft SharePoint implementations are not integrated with other parts of the company. The Connector for Microsoft SharePoint Document Libraries breaks down the walls between the implementation points by bringing all of the information about the content into a central location, which provides the infrastructure and integration that is necessary to repurpose content to serve multiple purposes and for consumption by multiple audiences from a single instance of the document. This is done without any new actions by the users, so that their daily tasks are unchanged by utilizing this integration. Additionally, this information can then be used in more complex business automation than is provided by native SharePoint workflows. Finally, the documentation can be records managed.

4.6.2 Architecture

Between version 2.1 and 2.2 of the SharePoint connectors, there were significant architectural changes in the product architecture due to functional enhancements around high availability and load balancing. Figure 4-11 on page 102 shows the new architecture.

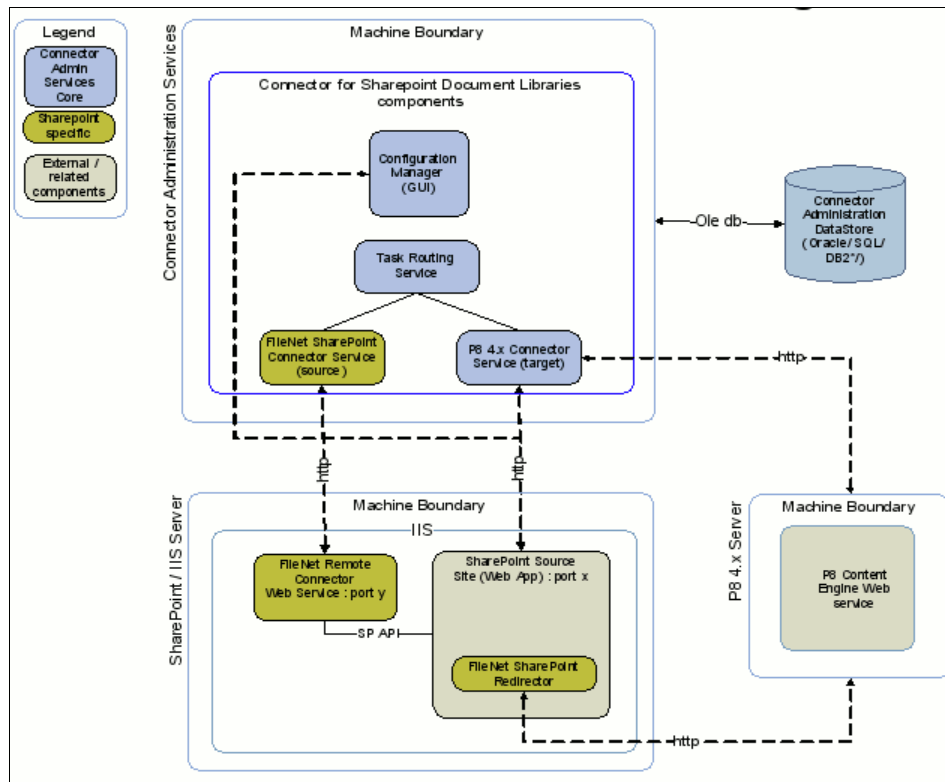


Figure 4-11 IBM FileNet Connector for SharePoint document libraries architecture

One of the most significant architectural changes between version 2.1 and 2.2 is the communications between the connectors and the SharePoint environment. In Figure 4-11, notice that with the exception of the Connector Administration database (which manages configuration and security policies for the connector), all of the communication is through HTTP. The component in the architecture that handles direct communication with the SharePoint server through the API is a FileNet Remote Connector Web service, shown in Figure 4-11, on the same machine as the SharePoint server. This new service combined with the changes in the communication method removes the dependency for the connectors to physically reside on the same machine as the SharePoint server so that farm and clustered environments can be supported.

Another advantage of the architecture in Figure 4-11 is that processor utilization can be balanced and controlled by removing the connector from the SharePoint environment and running it on a dedicated machine.

4.6.3 Services and integration points

The IBM FileNet Connectors for SharePoint Document Libraries is currently supported on the Windows platform. When the product is installed, there are three main components that can be distributed onto different physical machines or co-located on the same machine:

- ▶ The core Windows services
- ▶ The Remote Connector Web Service
- ▶ The SharePoint Redirector Web service

The core Windows services

The core Windows services consist of four Windows services that act as the core engine for IBM FileNet Connectors for SharePoint Document Libraries.

Figure 4-12 shows the services.

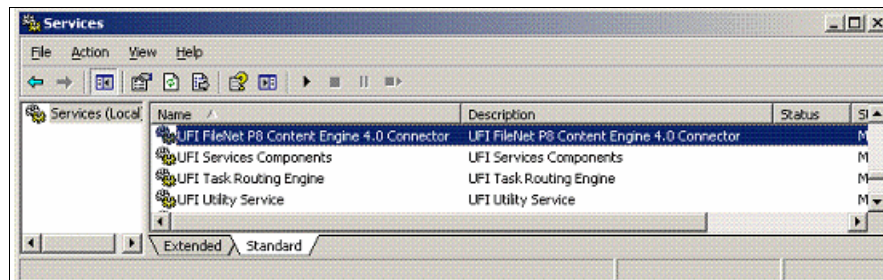


Figure 4-12 The core Windows services

The main service, which acts as a broker for the others, is the UFI Task Route Engine. When this service is started, it automatically starts any of the other services that are required. When it is stopped, it also stops the other services. The UFI Task Route Engine typically starts the UFI IBM FileNet P8 Content Engine 4.0 Connector, which is responsible for all of the communication to the back end content store and the UFI Utility Service that communicates indirectly with SharePoint through the Remote Connector Web Service (discussed in the next section). All of these services have comprehensive logging capabilities that can be enabled within the administration interface.

Note: The only service that is not governed by the UFI Task Route Engine is the UFI Services Components. This service is only required for processing the import and export of Task Route configuration and can remain stopped until this activity is performed.

The Remote Connector Web Service

From version 2.2 and higher of the Connector for SharePoint Document Libraries, a remote connector Web service is provided to perform the required automated tasks within Microsoft SharePoint. The UFI Utility Service, which we previously described, communicates through HTTP to the remote connector service, and if documents need to be retrieved or deleted or if links to documents need to be created within SharePoint, these operations perform the service. For more information about the SharePoint operations that the remote connector service provides, read the service description that is shipped with the product.

The SharePoint Redirector Web service

In some circumstances, it might be necessary to leave a link within a SharePoint site when a document is moved into the FileNet content repository. The Remote Connector Web service performs this operation. When a user within SharePoint clicks on a document link, the FileNet SharePoint Redirector Web service is called so that the system retrieves the document from the FileNet repository. The FileNet Redirector Web service uses a mapping within the Web.conf configuration file to obtain the name of the FileNet Content Engine. Using this mapping means that there is a level of flexibility to change the name of the FileNet content engine server without breaking the document links created within the SharePoint sites. For more information about how the content engine mapping is configured, review the installation guide and product help documentation.

4.6.4 IBM FileNet Connector for SharePoint Web Parts

Using IBM FileNet Connector for SharePoint Web Parts you can combine SharePoint's familiar interface with the content, process, and compliance infrastructure that the IBM FileNet P8 Platform provides. Users access and interact with IBM FileNet P8 object stores through one or more of the IBM FileNet Connector for SharePoint Web Parts. The list of the standard, out-of-the-box Web parts to connect to the FileNet content and process capabilities are:

- ▶ Content Editor Web Part: Use for formatted text, tables, and images.
- ▶ Form Web Part: Use to connect simple form controls to other Web parts.
- ▶ IBM FileNet Basic Search: Allows searching for document in a IBM FileNet P8 repository.
- ▶ IBM FileNet Advanced Search: Allows searching for documents in a IBM FileNet P8 repository.
- ▶ IBM FileNet Browse: Allows browsing through a FileNet Content Engine Object Store.
- ▶ IBM FileNet Personal Inbox: Displays the contents of the Inbox queue.

- ▶ IBM FileNet Public Inbox: Displays the contents of public queues.
- ▶ IBM FileNet Search Results: Displays results when searching a IBM FileNet P8 repository.
- ▶ IBM FileNet User Administration: Provides administration capabilities for users of FileNet Web parts.

The Web parts are added to SharePoint pages in the same way that the standard Web parts that Microsoft provides are added. They are listed in the Miscellaneous section. The Web parts are also designed to have a similar look and feel to the standard SharePoint Web parts. The way in which the documents and folders are listed is very similar to SharePoint, and despite the fact that the Web parts are accessing FileNet directly, the styling, column sorting, and right click menus options are tailored to look and feel like SharePoint.

The Inbox, Browse, and Search Web parts are the most typically used interfaces. The user can browse over content that they are authenticated for in the FileNet Content Engine security, which includes content that was created outside of SharePoint and content federated in from other locations.

Search results are similarly secured. The Web parts also fully support single sign on (SSO) either through the separate credential store database, which is provided or utilizing Kerberos support within Active Directory. To operate using the Active Directory method, it is assumed that users will logon to SharePoint using their domain accounts.

The Public and Personal inboxes are a key integration point. SharePoint users can directly view and interact with workflow task items in FileNet without ever leaving the SharePoint environment. This convenience means that users can employ more complicated automation with business critical operations and interactions. Management can use tools, such as Process Analyzer and Process Simulator, to respond, predict, and optimize these workflows, improving the business bottom line.

4.6.5 Summary of IBM FileNet Connectors for SharePoint

IBM FileNet Connectors for SharePoint provide a powerful content, process, and compliance infrastructure to ensure that Microsoft SharePoint activities are accessible to authorized users and placed under full life cycle and compliance management.

The internal architecture of the IBM FileNet Connectors for SharePoint gives it the flexibility and performance to provide enterprise level support for team collaboration. Users can enjoy the Microsoft environment with the extension of

FileNet's business process and compliance without ever needing to learn new skills or leave SharePoint.

4.7 Summary

The connectors and federation expansion products for the IBM FileNet P8 Platform provide a variety of ways to incorporate content from assorted sources. The content does not necessarily have to be moved from its original location, which allows corporations to maintain their existing systems. Content can also be migrated in an automated system. In either case, content can be classified, integrated, controlled, and reused in new ways, enabling businesses to make greater use of these important assets.



Expansion products for application framework

When information is readily available to use, users need applications that access and manage the content. In this chapter, we discuss the application framework related expansion products that enable business to quickly create and adapt applications to meet changing requirements.

The topics that we cover are:

- ▶ 5.1, “Application framework products overview” on page 108
- ▶ 5.2, “Electronic forms (eForms)” on page 109
- ▶ 5.3, “Business Process Framework” on page 114
- ▶ 5.4, “Business Activity Monitor and Cognos Now” on page 124
- ▶ 5.5, “Summary” on page 128

For an overview of all of the IBM FileNet P8 expansion products, refer to 3.1, “Expansion product overview” on page 56.

5.1 Application framework products overview

Application frameworks for enterprise content management enable businesses to develop applications quickly by providing a foundation of tested, enterprise-level functionality for broadly used applications. The expansion products that provide the application frameworks are:

- ▶ Electronic forms (eForms)
- ▶ Business Process Framework (BPF)
- ▶ Business Activity Monitor (BAM) and Cognos® Now

eForms, Business Process Framework, and Business Activity Monitor, each increase productivity by minimizing the amount of time and cost that is required to build user friendly applications. eForms integrates business processes seamlessly with an intuitive user interface that easily replaces and expands paper forms. BPF provides a configurable framework for business process management applications for commonly used scenarios, particularly in case management. Businesses use BAM to gain real-time visibility into their business processes. They become more agile, automating escalations and preventive actions.

These IBM FileNet P8 application frameworks focus on bringing both sophisticated functionality and simplified usability to users with a business rather than a technical background. The maturity of these products decreases project implementation risk, while adding flexibility. The benefit of such frameworks for IBM FileNet P8 customers are:

- ▶ Lower development costs
- ▶ Decreased project implementation risk
- ▶ Flexibility
- ▶ Reduced time-to-market
- ▶ Reduced time-to-value
- ▶ Ease of use

These frameworks deliver proven value. They arose out of market demand and IBM FileNet's long years of experience in delivering business value.

In addition, because this functionality is based on the IBM FileNet P8 Platform architecture, these benefits include all of the inherent functionality of the IBM FileNet P8 Platform technology:

- ▶ Audit trails
- ▶ High performance transaction oriented BPM
- ▶ Reporting
- ▶ System management

5.2 Electronic forms (eForms)

Electronic forms allow rapid development of intuitive user interfaces with quick response to market changes. Electronic form products were designed to replace paper-based forms and custom Web development to provide a single, flexible platform with which to rapidly develop user interfaces for business applications. The IBM FileNet P8 Platform supports electronic form technology with the IBM FileNet eForms and Lotus Forms. In this section, we discuss the architecture and use of FileNet eForms and briefly discuss Lotus Forms where appropriate.

Many applications require data entry and validation and become part of processes that range from simple document approval scenarios that simply require comments and decisions to complex applications forms with calculation fields, user interaction, security, and data lookups.

FileNet eForms comes with a Designer tool, and enables key functionality, such as electronic signatures, business process integration, and data lookup and validation. Additionally, as part of the IBM FileNet P8 Platform, these forms become part of business processes, automating, and streamlining work. This enterprise-level tool enables businesses to quickly transform cumbersome paper forms into fully interactive eForms that directly connect to the applications that drive business.

FileNet eForms Designer tool

Business users, architects, and other form designers create and manage FileNet eForms using the IBM FileNet eForms Designer. This graphical tool is a Windows-based application for creating, dragging, and configuring many different types of fields onto a form in much the same way as you do using a desktop publishing package, as shown in Figure 5-1 on page 110. The forms are stored as form templates.

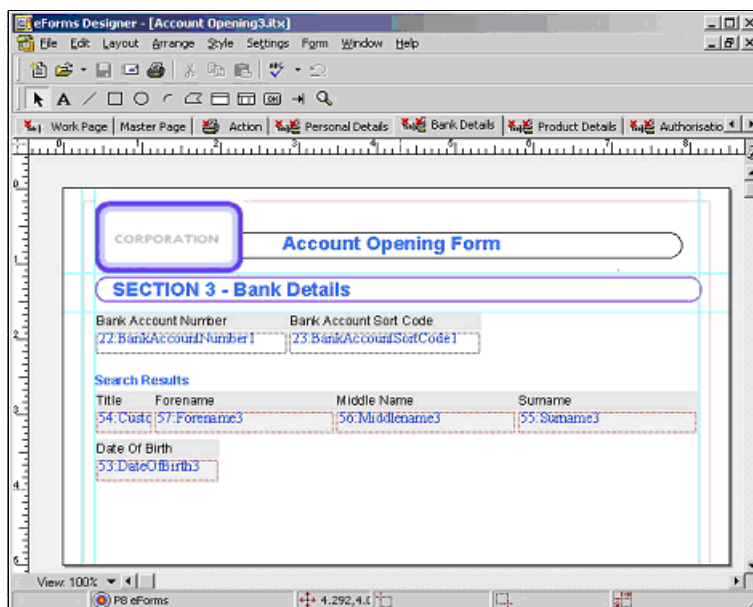


Figure 5-1 The FileNet eForms Designer application

Editors can create forms that are electronically identical to paper forms with tight graphical control, field layout, and ordering, and the appearance is all controlled here. Field content is also designed within this tool, from presentation to calculation and validation.

Electronic signatures and data lookup are two key features of electronic forms. Electronic forms often have unique capabilities over their paper-based cousins by allowing users to digitally sign areas of the form and transmit this information over a network, which is much quicker than paper-based delivery and cheaper than creating a scanning and indexing operation. Electronic forms also save paper storage. Electronic forms can also lookup and validate values for fields in the form, for example, a customer can type in their customer number and consequently their address information can be automatically retrieved from the corporate databases, which is a great time saver for the customer and greatly improves accuracy for the company.

5.2.1 Architecture

FileNet eForms and Lotus Forms make full use of the underlying IBM FileNet P8 Platform's ECM and BPM functionality, as shown in Figure 5-2 on page 111, which shows that the Workplace and WorkplaceXT applications are responsible

for determining how Web-based user and thick-client applications interact with form definitions and data.

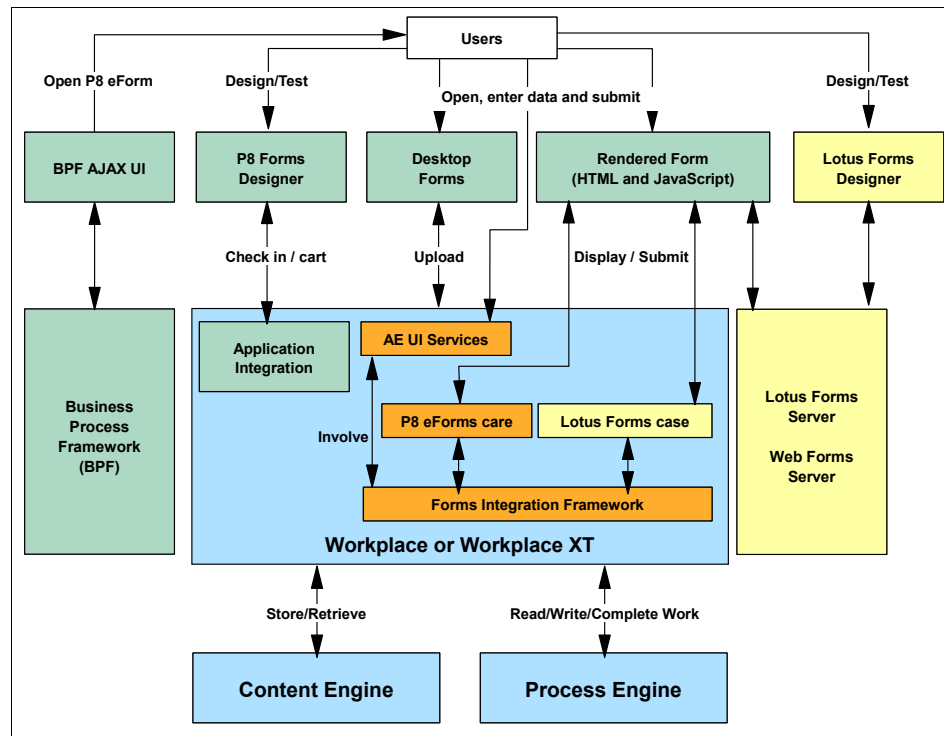


Figure 5-2 eForms architecture as supported by the IBM FileNet P8 Platform

Figure 5-2: This diagram focuses on the integration between user applications and the internal workings of Workplace and WorkplaceXT in regards to FileNet eForms and Lotus Forms. It does not expose background integration, such as BPF integration with the Content and Process Engines.

These forms templates are stored in the Content Engine like any other document. After it is stored, an administrator can create a form policy to instruct IBM FileNet P8 as to what to do with the data when a user populates a form, thus creating a form data object (a completed form). The form data object itself can be saved as a document and linked back to the original template. A business process can be launched in addition to or instead of storing the form. These form data objects and processes can use field data captured on the form. How that data is used and exchanged is specified in Form Policies or Workflow Subscriptions. Form data objects can be opened later and displayed exactly as they were filled in by the user.

This platform capability of linking a form template to a content repository and BPM suite was recently extended to allow Lotus Forms to also be used in the same way, which enables customers already using Lotus Forms to continue using the same products that they are used to while harnessing the power of the IBM FileNet P8 Platform.

An abstraction layer called the Forms Integration Framework handles the underlying concept of form data classes and form definition objects. When a user clicks on a completed form in Workplace, it is this library that determines which forms product was used to create that form. This library then invokes the relevant form's plug-ins to render the form itself. Additional configured options, such as form window size and title, submit and cancel buttons, and optional side panes (to show instructions, for example), are also rendered in the same window.

This standard mechanism abstracts the concept of form templates and form data objects to decouple the product from the IBM FileNet P8 Platform, which makes it easier to manage because administrators only have to learn core forms concepts, such as definition classes, data documents, and form policies, to work with either forms product.

The Lotus Forms Web application is necessary to render and perform sophisticated functionality with Lotus Forms. It is this Web application that provides complex data lookups, validations, and dynamic loading of sub forms to the displayed Lotus Form. This is where the form logic is handled. IBM FileNet P8 is used only as a form storage layer with IBM FileNet Content Manager and integration layer with IBM FileNet Business Process Manager. The FileNet eForms library provides similar functionality of lookups and validation for any rendered FileNet eForms. For either form technology, communication between the client browser and the Workplace server is handled through HTTP or HTTPS.

5.2.2 Integration and protocols

IBM FileNet eForms has the added benefit of being integrated with the Business Process Framework application. Normally in BPF, the case tab is a very simple grouping of textual fields that are shown within a grid box, which can be configured instead to use an eForm instead of the basic case tab. This creates a consistent method of user interactions throughout an application.

Thick client Windows applications, such as the FileNet eForms Designer application and FileNet Desktop eForms, are supported using communication either through the Application Integration jsp files or Workplace Web pages.

Desktop eForms is especially useful for cases that involve taking a form policy and definition offline, for instance, when there is no network connection. A good example of this is that of an insurance claims assessor who goes to customers

and assesses the damage to a vehicle after an accident. After the user returns to their office, they can upload these completed forms using Desktop eForms to Workplace. They are brought online and processed like any other form.

5.2.3 Customization and integration with third-party applications

eForms can be invoked directly from third-party applications by using the Application Engine User Interface (AE UI) Service, which is a URL based mechanism for opening a form definition for a user to populate. This URL can optionally pass initial data into the form. A return URL can be included with the request, which means that when the form is submitted into IBM FileNet P8 Workplace, the user's browser can be sent to a specified URL with information about the unique identifier of the form data document in the system.

This makes plugging eForms into a custom developed application very simple and shields application developers from any internal changes to the IBM FileNet P8 eForms functionality by providing a standard URL-based interface. Business Process Framework uses these APIs to integrate with eForms. As shown in Figure 5-2 on page 111, the arrow from the BPF AJAX UI points towards the user, which means that the HTML page has an internal iframe tag that calls the Application Engine UI Service. It is therefore the client browser, not BPF, that interacts with Workplace to load a form as a case tab. After this occurs, BPF uses the JavaScript™ API to read and write form fields. This all happens within the user's browser.

IBM FileNet eForms also provides a rich JavaScript API to allow developers to create unique value-add solutions. A common application of this flexibility is to use the JavaScript to create wizard-driven forms that guide the user step-by-step through a process.

Figure 5-3 on page 114 shows a navigational banner jsp page, displayed on the right side of the form, which shows the pages that are available in the form. Pages in the navigation can be added or omitted based on the user's interaction with the form.

Figure 5-3 A wizard-driven form created using the eForm JavaScript APIs

This information can then be displayed in the summary page. This makes the whole user experience much more intuitive for the user and more accurate at the same time.

5.2.4 eForm summary

The IBM FileNet P8 Platform's built-in support for IBM FileNet eForms and Lotus Forms provides a very quick and easily extensible method for creating a rich user interface for business applications. Electronic form support can be leveraged to create intuitive and very efficient user interfaces to support business needs. They are quick to develop, easier to maintain than custom Web pages, and pluggable into third-party applications.

IBM FileNet eForms also uses the same underlying functionality of the IBM FileNet P8 Platform to provide a standard document class and metadata model and enforce it. The IBM FileNet P8 Platform also enables speedy creation of business process applications and uses eForms as a rich interface with a very rapid time to market.

5.3 Business Process Framework

IBM FileNet Business Process Framework (BPF) reduces process application development cost and time to market and improves usability by providing a configurable framework for BPM applications. Many corporations have the same base needs for some applications: processes that are long-lived, requiring

multiple interactions. These requests might require review and approval. Document attachments and a log of these interactions are also needed. These and other requirements are basic functionality for case management systems.

The Business Process Framework is built right on top of the IBM FileNet P8 Platform, with out-of-the-box functionality, particularly for case workflows, inboxes, workflows, and UI design. This out-of-the-box functionality allows the business to focus on configuring rather than coding, which greatly accelerates application development.

Roles and Inbaskets

In Figure 5-6 on page 119, a user with the role of Administrator logged into BPF. This user's role shows three *inbaskets*, and cases in that inbaskets are shown on the right. A user can have multiple roles that are granted to them based on their job role or group membership. Each role has a unique user interface layout that provides the right information to them to perform that role. Each role might also have one or more inbaskets. An inbasket is simply a view of work to be completed within the system, which extends the concept of queues by allowing the BPF designer to restrict what queue items appear in the inbasket by providing a queue filter. Figure 5-4 on page 116 shows a sample window with user roles and inbaskets.

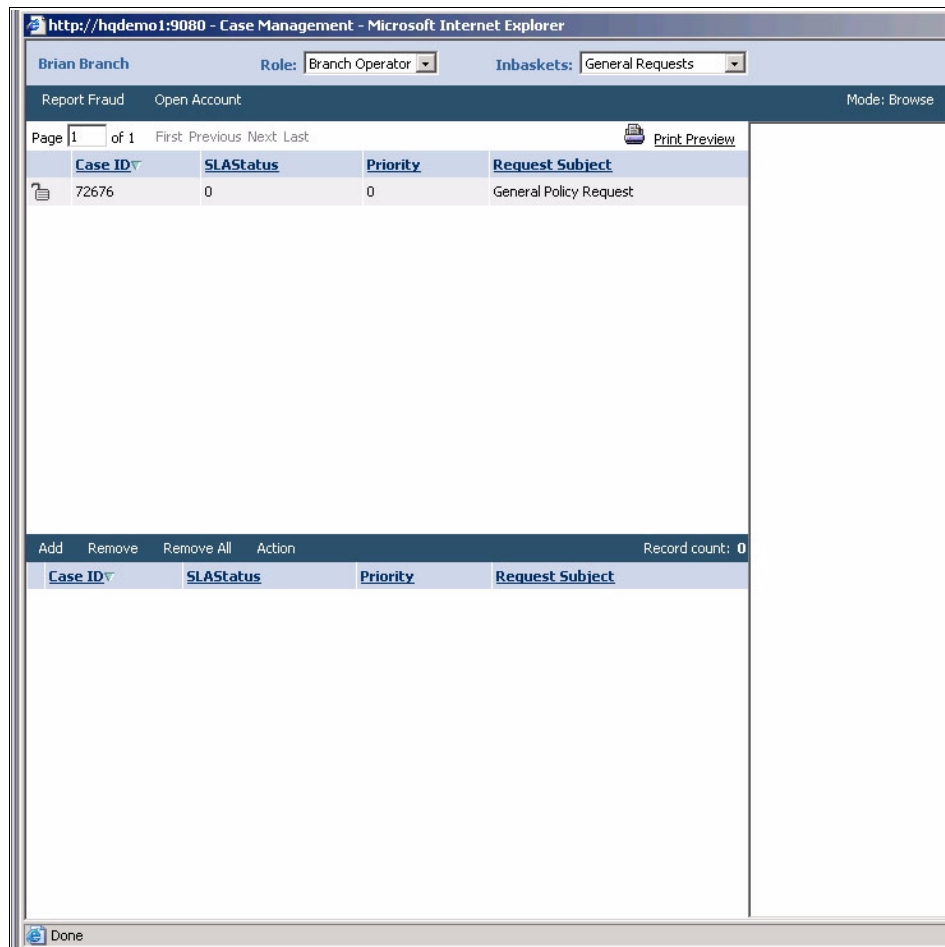


Figure 5-4 Sample user window with different user roles and inbaskets

An example of roles, permissions, and inbaskets might be a role of fraud investigator. All work for those investigators is placed in a Fraud Investigations queue. There might be junior members of the team that can only work on customer requests that are worth less than \$200,000. Senior team members can work on fraud cases of any amount. In this example, a queue filter for the junior fraud investigator inbasket restricts the view of work to only those items that are worth less than \$200,000. Additionally, junior members might be enabled to perform a certain subset of the process responses, for example, junior fraud investigators might not be able to perform the “Send to FBI” action; instead, they are only allowed to send the case to their supervisor.

BPF Explorer

BPF also comes with its own configuration tool, BPF Explorer, which allows administrators to rapidly configure, rather than code, a case management application. As shown in Figure 5-5, BPF roles, inbaskets, queue filters, case types, and lookup fields can all be configured within a single interface. The BPF Explorer is used to create an easily modifiable environment without requiring programming skills and enables real-time changes in a production environment.

In BPF Explorer, the links between the case, data, and BPM workflow are configured, which gives BPF the correct responses and steps to handle in the cases. In Figure 5-5, the properties for the inbasket for Pending Approvals is open, and a wide variety of information about the inbasket can be configured, including roles, responses, fields available, filters, toolbars, and tabs. For more information about case behavior management, see BPF classes and documentation.

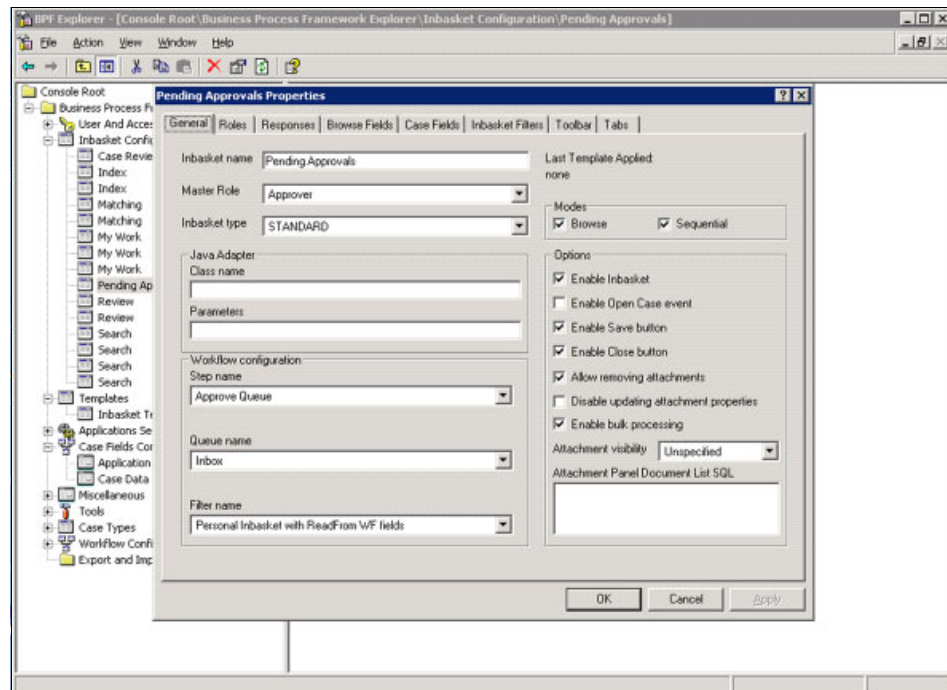


Figure 5-5 BPF Explorer with Inbasket properties

BPF Layout Designer

The BPF Layout Designer is a new visual design tool that permits the application architect to replace the default layout of the BPF Web Application user interface by creating new Layout Objects. Layouts are then assigned to one or more user

roles so that different layouts can be created for each role. It supports drag-and-drop changes with an intuitive interface, making Web design quick and simple.

The BPF Layout Designer is accessed through the Edit Layout action for users with permission for that action. In Figure 5-6 on page 119, different modules were dragged and dropped onto various portions of the UI to create a new user interface. Many of these modules are configurable from there, such as adding the logo location address. For more information about using the BPF Layout Designer, see BPF classes and documentation.

Case

Long processes with many interactions generate a large amount of data as either simple values, such as a house valuation, to more complex data, such as a detailed inspection and survey document. All of this information, within the processes and content interactions, must be held together for the entire life cycle of the process. In traditional, paper-based methods of working a bank branch, offering a mortgage service might keep a filing system with a case folder for each customer. This case folder concept is very useful for long-lived interactions.

The Business Process Framework provides this case metaphor for electronic documents and data that is held within a business process. It provides mechanisms for creating different types of cases, each which might hold certain key properties. BPF links these case objects to the relevant documents, data, processes, and folders in the ECM repository. BPF also provides a standard user interface case working paradigm to enable the rapid configuration of case-based applications. Figure 5-6 on page 119 shows a sample generic case application in BPF.

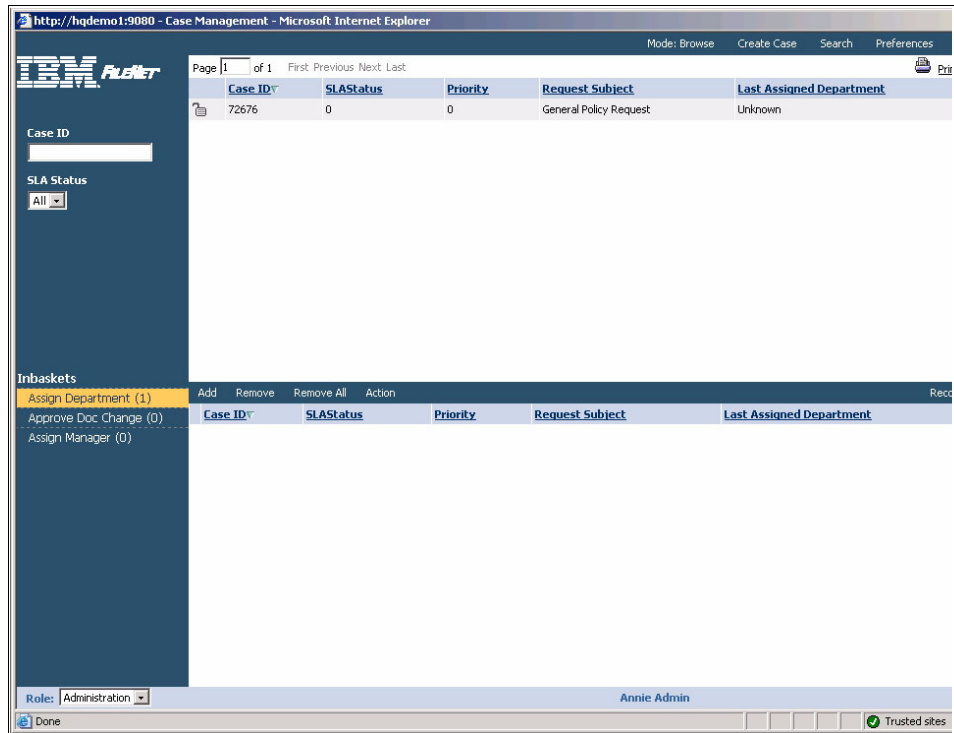


Figure 5-6 Sample generic case application in BPF

Case tools

Business Process Framework provides several out-of-the-box user interface features that can be configured and rearranged to rapidly create a customized case application. See Figure 5-7 on page 120.

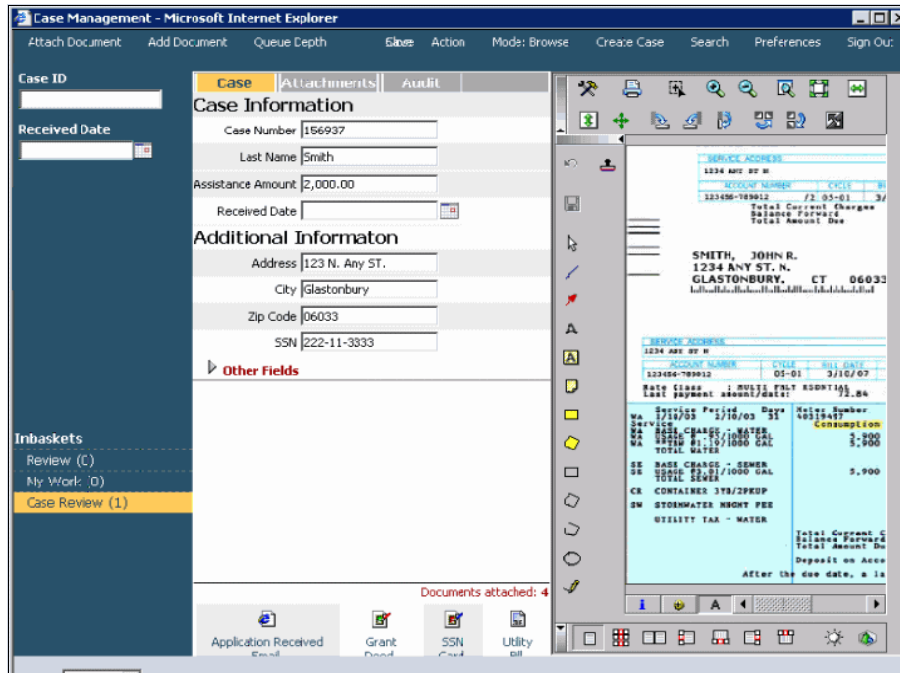


Figure 5-7 Case management interface

As you can see in Figure 5-7, there are several areas of the interface:

- ▶ Case view

A tabbed area of the interface where the main data of the case is shown, which includes a Case Information tab, Attachments tab, and Audit History tabs. Custom tabs can also be added.

- ▶ Inbasket list

The inbaskets can be shown either as a list or as a drop-down selection.

- ▶ Application toolbar

This area provides functions for the entire application, which includes user preferences, logout designer link, and search actions.

- ▶ Case toolbar

This toolbar provides the user with options that are applicable to either the inbasket view (when the list of work items is shown) or to the currently open case. These tools can be configured to include custom actions, such as links to eForms to fill out or customization to add a comment to the case audit history for collaboration.

- ▶ Logo
A logo element is provided to allow the organizations logo to be shown on the interface.
- ▶ Roles drop-down
A drop-down box is provided to allow the user to select which role they want to see the inbaskets and work for. Most users will have a single role, but this element supports multiple roles.
- ▶ External URL
A component is available to show an external URL, which can be a link to best practice information, google search, or any other interface that is useful to the user with the current role.
- ▶ Image Viewer™
The image viewer applet displays image files directly within the BPF interface. If this is omitted from the layout design, when an image is opened a pop-up window shows the image viewer instead. Having the image viewer shown next to the case fields is useful for work with rapid turnover, such as data entry or simple validation and routing to enable a user to quickly complete their work on a single window.

Business Process Framework also supports some more subtle user interface features:

- ▶ Case search inbaskets
Allows the user to search over their inbaskets to find cases with particular attributes.
- ▶ Case creation
Allows the user to create a new case, which can include adding attachments. Creating the case begins a new business process based on the configuration for this type of case.
- ▶ eForm case tab replacement
The case tab provides functionality to type in case data, view read only data, and restrict data choices to external lookups and drop-down options. It does not provide sophisticated calculation fields or validation. The case tab interface is a relatively simple single page of field names and values that are organized within a table. For more sophisticated user interfaces, it can be replaced by an eForm. eForms can be designed with calculation and validation fields, and have a very rich desktop publishing-like support for form design, which provides a very rich user interface option to replace the case information tab within BPF. Other tabs, such as Attachments and Audit, can be shown as normal and are unaffected by this feature.

- ▶ Access to ECM features

BPF was originally designed to present applicable content to process workers, not as an application within which to add content to an existing case. Thanks to the pluggable way in which IBM FileNet P8 Workplace and BPF are designed, it is possible to add BPF tools that access useful Workplace functionality in a pop-up window. Examples of this use include often used search templates or browse windows that allow the user to add existing content in the ECM system to the current case. This same functionality can also be used to open a new eForm for data entry or a document entry template for adding new documents into the system. This functionality is particularly useful for something like a bank branch worker where their work is a mixture of working on current cases and completing application forms on the behalf of customers who visit the branch. In this case, everything can be done from a single interface within BPF.

- ▶ Work sorting

Queue filters can be used to provide a default ordering to work items. This sorting can be used as priority, which can be calculated within a business process or if SLA timers and escalations need to be met.

- ▶ Push and pull working options

In some situations, it is desirable to force case workers to work on the next highest priority work item rather than let them pick and choose which work item to complete next. To support this, BPF inbaskets can be configured to get the next work item. When a user opens such an inbasket, instead of being shown a list of work items, BPF instead opens the first (highest priority) work item in this inbasket.

- ▶ Historical view of a case

BPF provides each case with its own audit history, which can be configured to log actions, such as who opened and saved the case and what actions they chose. A process can also log audit entries into the case audit history to provide case workers with detail into how back end systems or other processes interacted with a case. With a single case flowing through several discrete, independent processes, this audit trail can be very useful to gain an overall picture of what happened. Also, as processes do not leave a final record of outcomes, a case object can be used to show the final status fields for decisions and information that the process used, giving future users an insight into why decisions were made without complex reporting.

5.3.1 Architecture

Business Process Framework uses many of the underlying features of the IBM FileNet P8 Platform, adding only two new architectural elements, which are the

BPF configuration database and the BPF administrative and design Web application, BPF Explorer. The configuration database holds information about where case objects are created within the ECM repository, what queue filters to use, and inbasket configuration. The Web application reads this database to show particular user interface elements to certain users and to allow layout design using a Web interface.

Figure 5-8 shows the key architecture of the BPF expansion product.

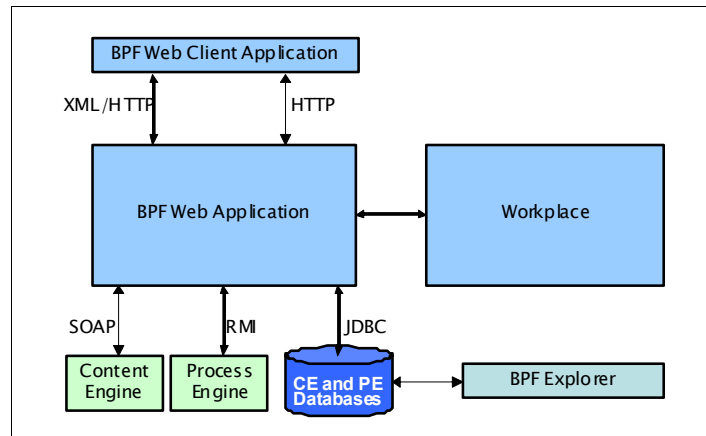


Figure 5-8 BPF architecture

All other information besides BPF configuration is held or accessed from the ECM and BPM systems themselves. The case objects, attachments, and audit entries are held as custom object classes within one or more IBM FileNet Content Manager object stores. The process responses, work assignees, and queue work lists are all held within IBM FileNet P8 BPM. As we mentioned earlier in this section, the eForms and Image Viewer functionality of Workplace is optionally used by BPF directly from the existing Workplace application. BPF also provides a Java Component that is installed into the component integrator to allow business processes to create, read, update cases and their attachments, and to add audit entries.

Although certain functionality is provided on the BPF Web interface, it is actually directly invoked by a Web browser. eForms, images, documents and folders are displayed by a Web page hyperlink open command from the Web browser on the client machine. The browser communicates directly with the Workplace, not BPF. This direct communication is achieved using the Web UI Service Commands.

BPF uses Asynchronous JavaScript and XML (AJAX) techniques to load information into the Web browser, which means that instead of redrawing the entire Web page when a link is clicked, like traditional Web applications, BPF

instead only requests the minimum amount of information that is required to complete its task and redraw only the parts of the interface that need changing, which minimizes the amount of information that flows over the network and makes BPF perform faster than traditional Web applications. AJAX interfaces are intended to make Web applications perform as fast as their desktop application cousins.

5.3.2 BPF summary

The Business Process Framework (BPF) provides a rich set of functionality on top of the core IBM FileNet P8 Platform to rapidly design, configure, prototype, and deploy process applications. BPF provides a rich, easy-to-use user interface. BPF also provides access to information in the underlying ECM repository, business processes, and external systems and presents this as a single, updatable case object. BPF is the perfect expansion product to enable IT and business analysts to rapidly create business applications without the custom coding and time scales that are associated with traditional business re-engineering projects. BPF has been used to create applications in customers that have gone from design to production in a short period of time.

5.4 Business Activity Monitor and Cognos Now

IBM FileNet Business Activity Monitor (BAM) is a Web based dashboard that observes and responds in real time to business activity. This functionality complements FileNet's other process optimization tools, which include Process Simulator and Process Analyzer (available with IBM FileNet Business Process Manager) by adding context to the information and relating it to high-level business requirements.

BAM is built on the IBM Cognos base product, "Cognos Now!", and expands the Cognos business matrix monitor to include IBM FileNet P8 specific tables, Key Performance Indicators (KPIs), and relationships with pre-configured variables and settings. Cognos was an external company that joined the IBM family.

Cognos Now! is also sold without the FileNet configurations as an appliance that combines both BAM and Cognos 8 BI reporting and analysis capabilities. The BAM product includes FileNet-specific configurations to accelerate time-to-value. BAM includes Key Performance Indicators and dashboards to monitor IBM FileNet P8 business processes. This out-of-the-box functionality saves an estimated six months of development time and configuration.

BAM provides visibility into business processes and performance and provides real-time event processing and alerts. It helps to identify issues for quick and

effective response, which enables both risk and cost reduction. BAM gives businesses greater control with customizable threshold notification and real time data modeling. Finally, events can be reacted to with automated responses including escalation and resolution that are initiated using the dashboard.

Users work with BAM in two main areas: the Dashboard and the Workbench. There is an additional appliance administrative tool as well. The dashboard displays data about the events that are being monitored, and the workbench is used to configure how those events are created along with other BAM administration.

We discuss these tools further in the next sections. For more detailed information about the full functionality of the dashboard, workbench, and appliance administration, refer to the BAM classes and documentation.

Operational Dashboard

BAM features a Web-based operational dashboard. Dashboard objects can be viewed, created, and altered by business users, not just by IT specialists, which enables the business to respond more quickly to market requirements. Views of information, including charts and alerts, can be viewed and managed in the dashboard.

These gauges, tables, and charts can be drilled down into to see additional details about the data. Dashboard displays can be configured for roles so that users have customized views of the business.

BAM Workbench

Because BAM is a dynamic modeling system, users can change their data models and apply them to live streaming data. Events can be associated with time and then correlated to show trends. The key difference between BAM and other monitoring products is that the data model is separate from a database schema and thus easily modified on the fly. By adding business plans and modes, a real-time update of how the business is performing against plan can be provided.

Most configuration settings for the FileNet Business Activity Monitor are performed from the Administration Console in the FileNet BAM Workbench. The BAM Workbench includes a Scenario Modeler and the Administrative Console and is accessed from a Web browser, just like the dashboard is. It can be used to configure rules, alerts, cubes, data source connections, views, events, and other BAM objects. BAM can be configured to notify users through a variety of options, such as e-mail, pager, and SMS.

For instance, fields, polling times, and other items that are relevant to specific events can be configured by selecting Events in the navigational pane. The

scenario modeler tab includes the ability to edit alerts that are based on business events.

Appliance Administration

This Web-based tool is used to administer the BAM appliance, which includes database sources. The Appliance Administration is where the connection to the Process Analyzer database is created. Information about runtime, memory, and agent status is also displayed here.

5.4.1 Architecture

BAM is integrated with IBM FileNet P8 so that real time business intelligence can drive business processes. Users monitor, initiate, and launch processes based on correlation and management of events, content, process life cycle, and task owner. It is integrated by accessing the data that the Process Analyzer produces, which it gained from the IBM FileNet P8 Process Engine event log.

Figure 5-9 shows how BAM interacts with IBM FileNet Business Process Manager's Process Analyzer in the context of the full business optimization suite. Note that the Process Analyzer is shown twice, as both producer and a consumer.

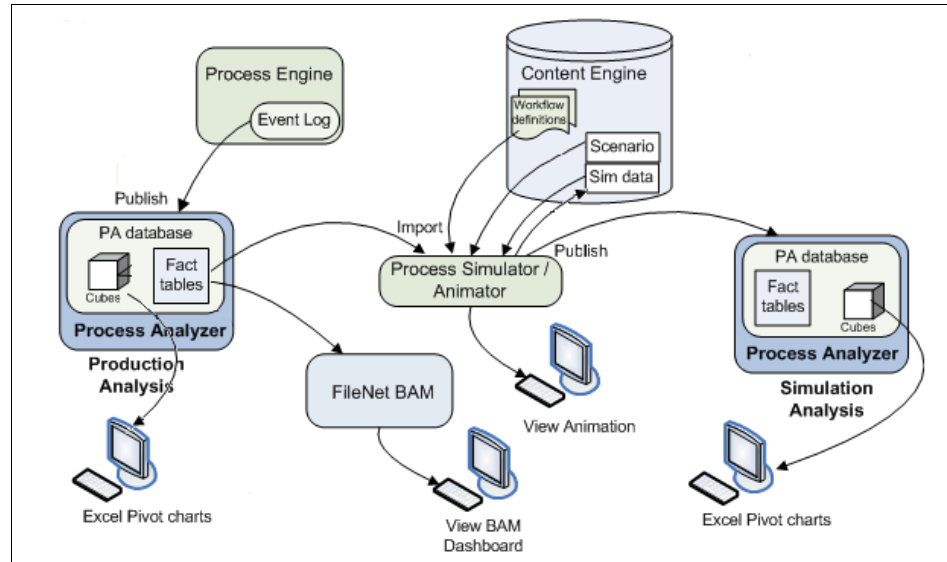


Figure 5-9 BAM interaction with Process Analyzer

In Figure 5-9 on page 126, the Process Analyzer consumes production event data from the Process Engine (left) and simulation event data from the Process Simulator (right).

The Process Simulator uses workflow information from the stored workflow definitions and production workflow statistics from the Process Analyzer (PA) to produce a scenario. The production workflow statistical data contains the actual step and workflow processing duration. FileNet Business Activity Monitor evaluates analytical data in the PA fact tables. Figure 5-5 on page 117 shows BAM architecture.

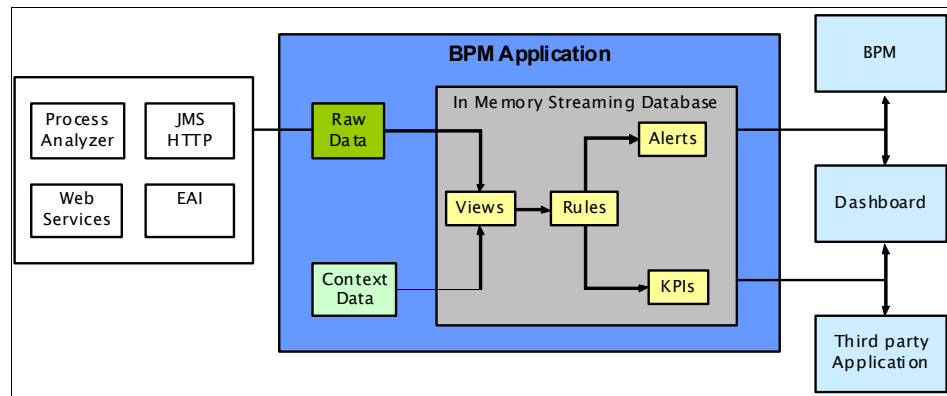


Figure 5-10 BAM architecture

Process Analyzer creates a database from information about running workflows (business processes). BAM retrieves its data from the Process Analyzer database Fact tables. The basic objects that are necessary to access the workflow and work item data and additional objects for more advanced activity are stored on the Process Analyzer. To use these objects, they must be imported into the FileNet Application Workbench in BAM. After they are imported, they can be used as-is or modified.

BAM can integrate with a variety of sources of background data to give businesses a clear contextual view of events. Events and data are stored in memory, in a streaming database. This event-driven approach enables BAM to notify users more quickly and to be modified more easily. Alerts in turn can drive new processes, completing the circle of information.

BAM also uses a metadata database for its own use, which contains the definitions of all objects in FileNet Business Activity Monitor installation. This metadata database also contains the details of alerts and object runtime data persisted to disk.

5.4.2 BAM summary

BAM empowers business users with control and access to critical operational information. Users can quickly respond to events as they occur or proactively monitor events. It gives greater insight into business performance real-time across multiple systems and functions.

5.5 Summary

Application frameworks for enterprise content management help businesses produce better applications more quickly and optimize their business processes. eForms, Business Process Framework, and Business Activity Monitor greatly reduce development time while delivering solid, industry-standard functionality. The tight integration each of these have with IBM FileNet P8's Process Engine ensures solid functionality, performance, and scalability. By using these products, users realize better time to value and greater user acceptance.



Expansion products for search, classification, and discovery

In this chapter, we discuss search and discovery products that work with IBM FileNet P8 Platform to make content more accessible and tightly controlled for legal and other business purposes.

The topics that we cover are:

- ▶ 6.1, “Search, classification, and discovery product overview” on page 130
- ▶ 6.2, “IBM Classification Module” on page 130
- ▶ 6.3, “eDiscovery Manager and Analyzer” on page 134
- ▶ 6.4, “IBM OmniFind Enterprise Edition” on page 137
- ▶ 6.5, “Summary” on page 139

For an overview of all of the IBM FileNet P8 expansion products, refer to 3.1, “Expansion product overview” on page 56.

6.1 Search, classification, and discovery product overview

Businesses need to be able to organize, find, control, repurpose, and eventually dispose of the content that they store in their repositories. The IBM search, classification, and discovery products address these concerns with the following offerings:

- ▶ IBM Classification Module
- ▶ IBM Content Analyzer
- ▶ IBM eDiscovery Manager and Analyzer
- ▶ IBM OmniFind Enterprise Edition

With these products, companies can proactively electronically store information with governance, in-house identification, collection, processing, culling, holding, and exporting.

Organizations can organize and classify documents with IBM Classification Module, find the content with IBM OmniFind Enterprise Edition, and provide records management with IBM FileNet Records Manager (the core IBM FileNet P8 product). IBM eDiscovery Manager and Analyzer combine and expand some of this functionality to address legal concerns for search, retention, and reporting according to regulatory requirements.

Using this tightly integrated technology organizations can proactively gain control over structured and unstructured electronically stored information with state-of-the-art archiving, classification, retention management, and analytics.

IBM Content Analyzer: We do not cover IBM Content Analyzer (formerly known as OmniFind Analytics Edition) in this book. For more information, refer to the following IBM Redbooks publication:

- ▶ *Introducing OmniFind Analytics Edition: Customizing Text Analytics*, SG24-7568

6.2 IBM Classification Module

IBM Classification Module (ICM) automates the organization of unstructured content by analyzing full text of documents and e-mails. IBM Classification Module can accurately and automatically organize information so that it is more accessible and more leverageable and also to help reduce your risk and costs.

Automating the classification of your content means that you can get more out of your ECM system, with less time invested.

ICM uses natural language processing to analyze the content of documents and e-mails to categorize them. It learns to categorize from examples in your enterprise or from keywords and combines classification by text analysis with rules.

A category is a specifically-defined division in a system of classification. A category is a label that is used to mark texts to indicate that they belong to a particular class of texts, and they can be content or an attribute of the object. Documents are associated with categories to create a Knowledge Base. A Dictionary defines the content type and data type of a field.

The Configuration Manager is an administrative application used to create, import, and export Knowledge Bases. It manages (start/stop) the Knowledge Base, modifies its properties, and modifies its dictionaries.

The Classification Workbench is an application that is used to work on a Knowledge Base and creates categories in the Knowledge Base based on the categories indicated in the training. Creating the taxonomy, training, and fine tuning classification is also done in the Classification Workbench. The Classification Review Tool is used to review the automated actions and set the proper level of automation. It also can configure which folders or documents to include or exclude and a filter by date.

ICM can learn in real-time, adapting its understanding based on feedback from users or administrators, which means that the tool can be trained to make better automated decisions. When manually ingesting sample documents, it can suggest folders to put the documents in, document classes to assign it to, and categories and property values. It reads and understands the full text of the document. It is trained at this point and can select other folders, document classes, or properties if desired. It is key that this training be done over representative data, in a representative distribution; otherwise, the training will be falsely skewed.

Content can be reclassified or added to new taxonomies when they are created. As new applications crop up that must reuse the existing content under management, and require new metadata, and new taxonomies, you can use this solution to generate that metadata and classify and reclassify the existing content.

6.2.1 Integration and connection points

ICM is integrated into IBM FileNet P8 for bulk classification of content upon ingestion or reclassification of content that is already under management. It is integrated in two parts: by configuring ICM remote server URLs to point to the IBM FileNet P8 server and by altering the object store. The IBM FileNet P8 server has metadata added through the IBM FileNet Enterprise Manager to the object store. An XML file that contains the Classification Module add-on information is configured in the Enterprise Manager, which is used to add the property definitions from the add-on to the object store's base document class.

Folders and document class names must be kept in sync between ICM and FileNet because change in one means the other must be changed to match.

Protocols

All client libraries are based on SOAP. ICM API supports the Java, COM, and C++ programming languages and defines the same basic set of structures/objects and functions/methods for each language. Full documentation of the libraries and their use is in the ICM Developer's guide.

6.2.2 Architecture

The IBM Classification Module is a distributed, scalable platform for providing Relationship Modeling Engine services. It runs a set of server-side processes on a single machine or multiple machines. It also provides a number of client libraries for remote access that are designed for various development environments. Figure 6-1 shows the data flow of the ICM learning process.

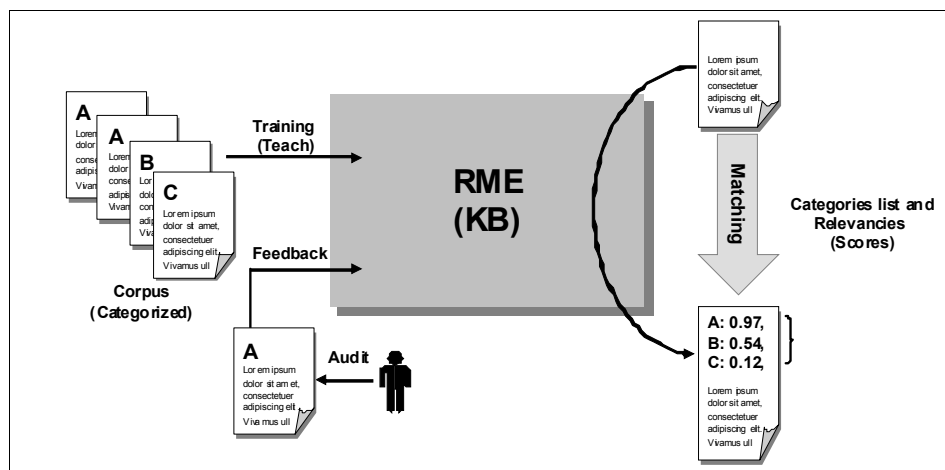


Figure 6-1 ICM learning data flow

To teach the knowledge base how to categorize the documents, a training set is loaded into the Classification Workbench, which is the application that is used to create and train knowledge base. Feedback given to ICM changes the way text and properties are evaluated and processed, which changes the category list and relevancy of the properties, resulting in better matching and a more useful Knowledge Base.

Figure 6-2 shows the ICM server architecture.

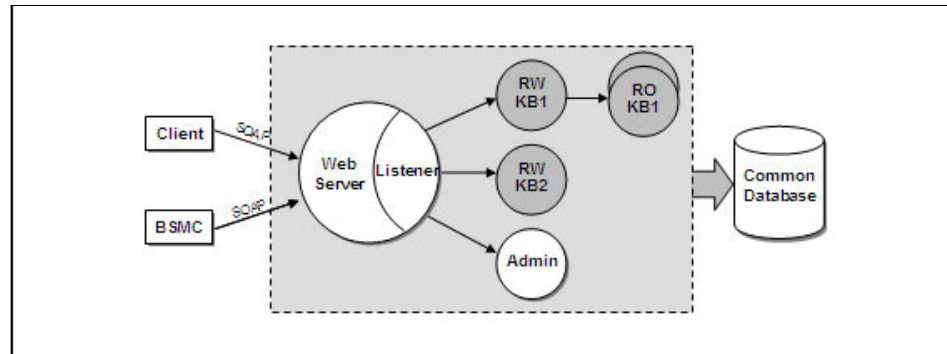


Figure 6-2 ICM server architecture

Clients communicate to ICM on the Web server through SOAP, which then communicates with the Knowledge Bases or the administrative tool.

The Web server is either IIS or Apache. The listener that is inside the Web server is responsible for routing requests to the appropriate components. RW KB 1 and RW KB2 stands for Knowledge Base Read/Write 1 and 2, and they are responsible for read and write requests on a specific Knowledge Base. There is one instance per Knowledge Base. RO KB1 is responsible for read only requests on a specific Knowledge Base. There can be multiple instances per Knowledge Base. The admin process is responsible for server administration requests. The Common Database is the persistent data store for configuration information, history, and other data.

6.2.3 ICM summary

IBM Classification Module is a platform for automating decision making in the enterprise content management architecture. Businesses need a standard and consistent way to understand and access the unstructured information that resides in their various file systems and content management repositories. IBM Classification Module (ICM) makes useful content searchable and found quickly resulting in faster business decisions that drive customer satisfaction and value.

6.3 eDiscovery Manager and Analyzer

IBM eDiscovery is an integrated solution that helps to control stored information and improves litigation response across the enterprise. Leveraging the scalable ECM platform, IBM eDiscovery enables companies to manage the entire eDiscovery process in a security-rich, audited, and defensible manner.

As only eDiscovery Manager is released at the time of writing, Analyzer is only be mentioned briefly.

eDiscovery Manager provides an automated hold and preservation process, first pass review, process automation, and a chain of custody to collect and preserve key electronic evidence. The company can manage litigation holds and retention on a case-by-case basis.

eDiscovery Manager can search, select, and analyze content for early case insight and reduce content volume for further legal review. It offers secure, audited collection and management of e-mail and other electronically stored information. Additionally, it provides proven auto-classification and robust records management to help IT departments manage information proactively for compliance and electronic discovery requests. It provides out-of-the-box searching, selecting, holding, and export tools coupled with industry-leading rich-content analytics tools that help compliance investigators and in-house counsel analyze, prioritize, and filter collected case materials to optimize case preparation and reduce volume and the cost of litigation review. The relevant e-mails can be exported in .pst and .nsf formats for further detailed review.

eDiscovery Manager integrates with IBM FileNet Business Process Manager to help organizations standardize, control, and automate legal discovery workflows and enable third-party components, as needed.

eDiscovery Analyzer takes the records that are built in eDiscovery Manager and is optimized for deeper records review.

6.3.1 Architecture

An eDiscovery Manager system includes the following main components:

- ▶ Web browser
- ▶ WebSphere Application Server
- ▶ An archive server
- ▶ E-mail client

Records management software is optional.

Figure 6-3 shows that the source machines connect to the task router through the Web browser, which connects with the task router through the APIs.

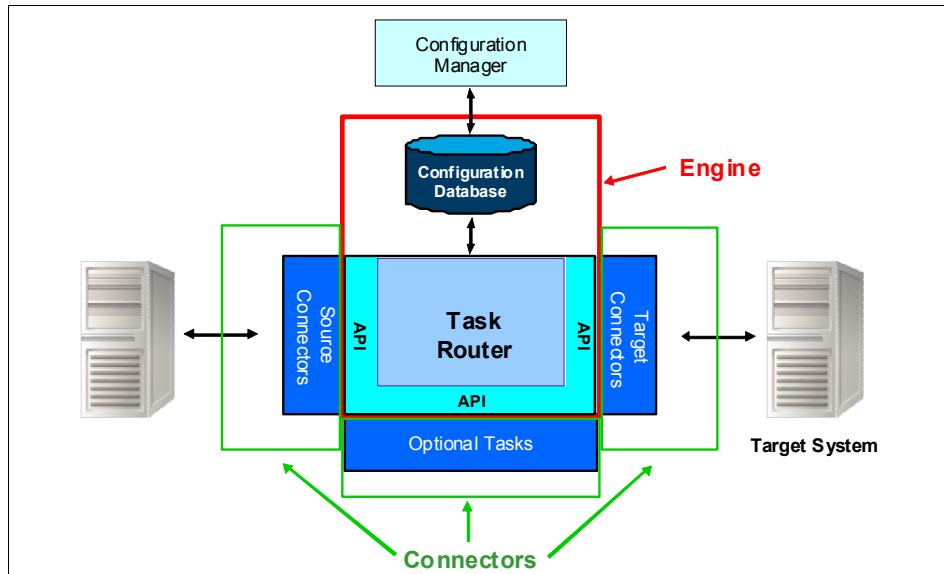


Figure 6-3 High level look at eDiscovery Manager

6.3.2 Integration and connection points

In this first release, eDiscovery Manager is integrated with e-mails and attachments that are archived with either IBM CommonStore or IBM Content Collector and stored in IBM Content Manager. eDiscovery Manager also retrieves e-mails and attachments that are archived with either IBM FileNet Email Manager or IBM Content Collector and stored in IBM FileNet P8.

Figure 6-4 on page 136 shows eDiscovery Manager integration with IBM FileNet P8.

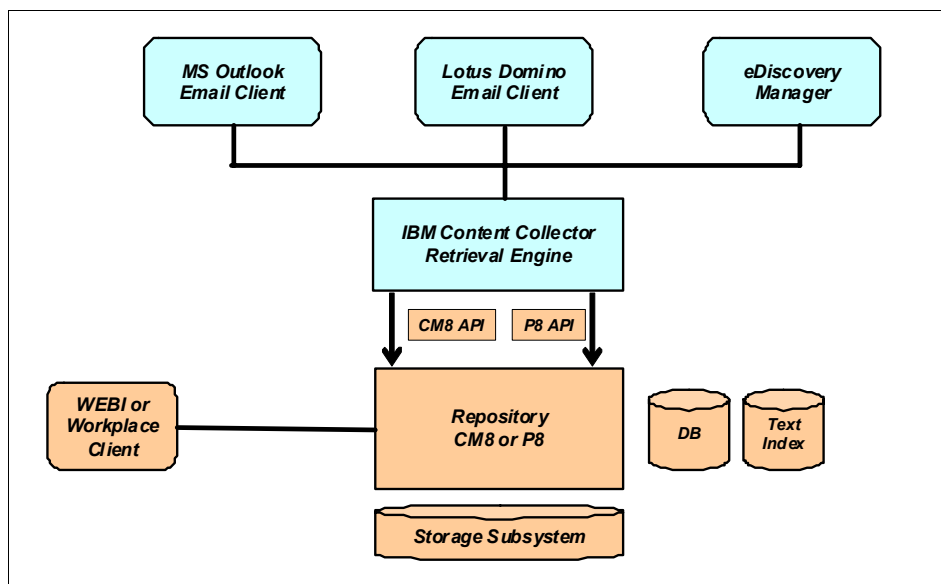


Figure 6-4 eDiscovery Manager as integrated with IBM FileNet P8

At the time of writing, when integrated with IBM FileNet P8, the e-mails' subject line is searchable with. Additional functions will be added in future releases.

eDiscovery Manager is fully integrated with IBM FileNet Records Manager. This is an optional configuration. E-mails that are placed in a collection are placed on hold, and this hold trumps any other records hold. In addition, explicit records manager holds can be placed directly from the eDiscovery Manager user interface.

For integration with business process management, these audit records, cases, and searches can be checked into IBM FileNet Content Manager to launch a workflow or embedded into a workflow for legal compliance, which is a manual integration.

6.3.3 Summary of eDiscovery

IBM eDiscovery is an integral part of the IBM Compliance Warehouse for Legal Control, which is the first integrated, secure, audited solution that combines software, hardware, and services in a unified environment to enable organizations to achieve, sustain, and prove compliance with multiple legal and compliance mandates while reducing cost, complexity, and risk. The combination of IBM eDiscovery and IBM Compliance Warehouse for Legal Control provides a complete solution for secure e-mail evidence preservation in an audited

repository of record with chain of custody preservation and integration with the IBM market leading records management.

6.4 IBM OmniFind Enterprise Edition

IBM OmniFind Enterprise Edition provides scalable, secure, high-quality enterprise search. It is designed to improve the productivity of knowledge workers and maximize the value of portal and collaboration investments. It provides a platform for constructing semantic search and content analytics solutions, such as entity analytics, sentiment analysis, threat analysis, global name recognition, and more. OmniFind Enterprise Edition provides a rich enterprise search and text analytics platform and delivers highly relevant and language-specific search results.

6.4.1 Integration and connection points

OmniFind Enterprise Edition has search and index APIs for creating custom search applications. These APIs can submit search requests, process and fetch search results, or browse taxonomy trees. They can also administer collections and enable indexes to be searched.

Crawler APIs are used to add, change, or delete information in the document or the document metadata and to indicate that documents are to be ignored (skipped) and not indexed. Search and Crawler APIs use HTTP or HTTPS protocols. The search APIs can get unified result sets over multiple collections if they use LDAP or JDBC access protocols.

Identity management component APIs search multiple repositories with a single query and views only the documents that are allowed by the appropriate security.

There is extensive documentation of the query syntax in the Programming Guide and API Reference.

6.4.2 Architecture

Figure 6-5 on page 138 illustrates the OmniFind Enterprise Edition communication architecture.

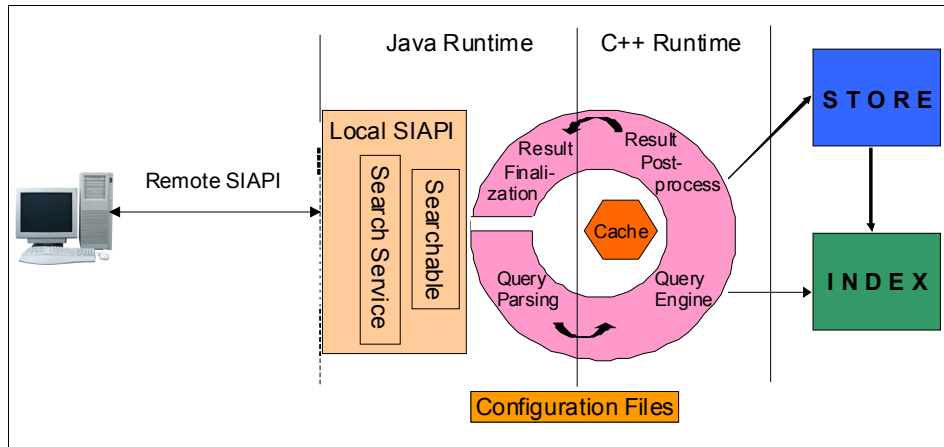


Figure 6-5 OmniFind Enterprise Edition communication

The search server stores the collection data for the enterprise search system. The Search and Index API (SIAP) Search Service keeps arrays of searchable objects. It initializes and refreshes these collections. The Searchable module keeps collection configuration data, which includes security options, tokenizer-related options and dictionaries (for example, synonyms, stop words), and other properties. The Searchable modules are also responsible for all search and count operations.

In the C++ Query Engine, the query is serialized over the socket from the Java runtime. The query cache is consulted, and if a hit is found, the evaluation is bypassed. Query terms are processed, and then evaluation takes place over delta and then main indexes. Candidate results are determined and scored and inserted into candidate heaps.

In the C++ post processing, the results are typically sorted by relevance, by date, none (by order found), or by field: numeric or string. Metadata is fetched from the object store. The results are then summarized, and the entire result set is inserted into the cache if it fits the cache.

6.4.3 OmniFind Enterprise Edition summary

Stored information is useless if the relevant data cannot be found easily. OmniFind Enterprise Edition is designed to improve the productivity of knowledge workers and maximize the value of your portal and collaboration investments by making that information retrievable and relevant.

6.5 Summary

IBM FileNet offers a suite of discovery and compliance products designed to help companies be successful meeting their legal requirements and making better use of their information assets. These powerful products expand the IBM FileNet P8 Platform to harness the content management facilities in important areas of control and security. The business process management facilities are augmented using IBM eDiscovery as content is made more available and better organized. This organization assists customers in meeting official requirements and regulations, which reduces corporate exposure and risk.

Summary of all expansion products

In this chapter along with the previous three chapters we discuss the expansion products for the IBM FileNet P8 Platform and how they integrate and add value to the IBM FileNet P8 systems. IBM FileNet P8 Platform provides a wide range of facilities that enable application developers to build compelling extensions products. These products make it easier to incorporate and make use of intellectual assets. Information is better organized, more accessible, and easier to use with these solutions.

In addition to the products that we mentioned here, there are over 200 partner line-of-business and technology applications that provide additional expansion beyond these capabilities, and they form a clear competitive differentiator and are a compelling extension to IBM FileNet P8's value proposition. For more information on partners and their offerings, contact your sales representative or refer to the IBM ECM Partner Solutions Handbook.

These applications and solutions, both IBM and partner, prove the flexibility and strength of the IBM FileNet P8 Platform. These products are also rapidly adapting for future capabilities with the delivery of the newest version of the platform, version 4.5. IBM FileNet P8 4.5 delivers flexibility in a services-oriented environment to help empower business users, shorten time to value, and respond faster to changing requirements. Also known as *agile enterprise content management*, it delivers solutions rapidly to solve increasingly complex business problems, helping organizations make better decisions faster, which includes features, such as simplified modes for process creation and a drag and drop iWidget facility for rapid application development.

Applications that are already taking advantage of IBM FileNet 4.5 include IBM FileNet Records Manager 4.5, IBM eDiscovery Manager, and IBM Content Collection.



Enterprise content management

In this chapter, we describe the advantage of an enterprise content management (ECM) system over a database-driven content repository. We discuss how metadata and access control help to activate content in the context of business processes and how the unique value that the event driven architecture of the IBM FileNet P8 Platform can be leveraged to tie content and processes together. In addition, we illustrate how the Content Engine and Process Engine can integrate with external systems in the context of a business process. On the example of records management and compliance, we show how the IBM FileNet P8 Platform design speeds up the implementation of appropriate solutions.

The topics that we cover are:

- ▶ 7.1, “Anatomy of an ECM infrastructure” on page 142
- ▶ 7.2, “Content event processing” on page 154
- ▶ 7.3, “Content life cycle” on page 163
- ▶ 7.4, “Business processes” on page 166
- ▶ 7.5, “Records management” on page 180
- ▶ 7.6, “Summary” on page 185

7.1 Anatomy of an ECM infrastructure

In this section, we discuss the typical functions that are required for a system that supports content management on an enterprise level, and we map the results to the IBM FileNet P8 architecture, which we described in more detail in the previous chapters. We discussed *Active content*, one of the most outstanding capabilities of the IBM FileNet P8 architecture, in 7.2, “Content event processing” on page 154.

Table 7-1 shows the definition of important terms that we use throughout this chapter.

Table 7-1 Definitions

Term	Definition
Process Engine server	A Process Engine server is a single server instance that runs the Process Engine software. In a virtualized environment, multiple Process Engine servers might be running on the same physical server.
Process Engine system	A Process Engine system is a compound of one or more Process Engine servers. The Process Engine servers can be either farmed or used stand-alone. A Process Engine server can only be the member of one Process Engine system, and in a single IBM FileNet P8 domain only one Process Engine system can be defined.
Isolated region	A Process Engine system can be partitioned into isolated regions for segregating data from each other. Data from different isolated regions is kept in different tables of the database.
Work object	A work object is a single instance of a process executed in a Process Engine system. Any work object is bound to exactly one region and cannot cross the border to another region. Work objects are represented as rows in the Process Engine database tables.
Process definition	The process definition, also referred to as process model or workflow map, is a representation that describes how the steps in the process are sequenced and related to each other.
OLAP	On Line Analytical Processing Database technology that uses multi-dimensional structures (OLAP cubes) to provide rapid access to data for analysis purposes.

Term	Definition
SLA	Service Level Agreement
KPI	Key Performance Indicator

7.1.1 Store and retrieve

At a first glance, content management might seem like a next-step evolution that extends the functionality of the traditional store and retrieves operations that an electronic archival system delivers. Indeed, effectively managing image data that does not change its content after it is ingested is one of the key features for an ECM system. This is due to the fact that any larger organization requires moving their paper-based processes to electronic images to gain efficiency that they could never accomplish when continuing to work with paper. We further elaborate on this topic in 7.4, “Business processes” on page 166.

For this purpose, all we need is a storage for files and a database system to keep track of which file is stored at which location to allow for a fast retrieval. All of the other functions that are intrinsic to modern content management systems, such as the ability to manage different versions of a document, keep track of who created a new version, and check-out and check-in mechanisms, to ensure integrity when collaborating on documents could be built on top of this architecture.

However, the ability to scale the system to meet virtually any customer’s requirement regarding the ingestion rate for new documents and a quick access to content objects under management is still key for an ECM infrastructure. Scalability is one of the aspects where solutions often fail as they reach a certain limit where the internal architecture does not allow enhancing the throughput any further, regardless of how much server power is provided.

Compared to the old days, when the memory and processing power were available on a server, which was one of the most limiting factors for the performance of an electronic archive, modern systems must allow scaling on both levels, vertical and horizontal to provide virtually any magnitude of resources needed. An ECM architecture must be able to leverage vertical and horizontal scaling to convert the resources provisioned by the IT infrastructure into performance of the solution. IBM FileNet P8 exactly meets this requirement because it allows scaling of the three core engines (Application, Content and Process Engine) vertically and horizontally. Refer to Chapter 9, “Scalability and distribution” on page 249 for details about the scaling capabilities for the IBM FileNet P8 Platform.

7.1.2 The case for metadata

When analyzing the way users gain access to a certain piece of content, it becomes apparent that in the beginning, you cannot determine which exact set of objects are needed. This ability only works if you store a unique identifier that is returned by the repository on the level of the applications that use this repository. After the content object is needed, it can be retrieved by passing the identifier back to the repository and requesting the content for it.

If you do not store this identifier on the level of the application, or you need to check whether other content that is in some relation to the context you are currently working on exists, a way to search for objects must be established. To allow for efficient searches, content that is stored in a repository must be accompanied with additional information that allows finding particular objects or distinguishing different pieces of content that were stored. For this purpose, *metadata* must cover common attributes that are maintained at system level (for example, who added a certain document to the system and when it occurred) for all objects and individual information, which is only used for a limited group of objects that belong to the same type (a vendor number for invoices).

Let us consider a file system on a computer, where the only custom metadata attribute commonly available is the file name and the element of a folder for structuring. Because the limitation of short filenames disappeared, this often leads to very long and complex file names that are the result of an aggregation of additional information for that particular piece of content (for example, “letter to customer xyz - contract 123-456.doc”). It is apparent that based on this structure a system can never efficiently handle a request, such as “show all contracts for customer xyz”, whereas this is an easy task after important criterion to search and distinguish objects are stored as metadata. Additionally, the segregation of objects into different classes that have different metadata properties adds another degree of freedom for efficiently structuring content objects in a repository.

The requirement for metadata does not only apply to content objects but also to process instances. Again, system-related information, such as the launch date for a process or instance-specific data, such as a vendor number must be available for process instances. Based on metadata, process instances can be filtered in applications, such as work inbaskets, to efficiently gain access to the piece of information needed, for instance, consider the scenario of a call center where agents must quickly locate the process instances that are related to customers calling in to check for the status of their requests. Of course, it must also be possible to search for process instances (often referred to as *work items* or *work objects*) based on metadata information.

The IBM FileNet P8 Platform supplies rich support for storing metadata together with content elements *and* process instances and searching for content or processes based on their metadata. For content elements, the Content Engine API provides the ability to search across multiple Object Stores and to return the results in a consolidated list. With this search ability, you users can find a content element, even in cases where it is not known in which Object Store the content element is stored. Work objects are maintained in queues, and the Process Engine API not only provides methods to search for a work object within a queue, but it also has the notion of a *roster* that tracks the location of all work objects in an isolated region and can be used as a starting point to find work objects for which the current queue is not known.

Workflow subscriptions define how workflow fields are mapped from content fields when a piece of content triggers the execution of a workflow without any coding. In the next section, we describe in more detail which metadata capabilities are important in the context of ECM.

7.1.3 Enterprise catalog and content federation

Even though it might be desirable to have only one single content management system running within an organization, it is common for large companies to have multiple content management systems from different vendors because either content management and image management started as departmental solutions at different units across the enterprise or because mergers and acquisitions caused heterogeneous infrastructures.

In theory, for many content management systems, their content can some how be converted and migrated into another vendors system. However, because APIs for content management systems are vendor specific and no common API exists, it is also required to re-implement the applications or at least the functionality that they provided, on top of the new system into which the content was migrated. Whereas this migration is again possible for many applications, the cost/benefit analysis might show that the ROI is achieved only after years, especially for very complex bespoke implementations.

In general, the adoption of SOA as a design pattern might help, as it introduces an abstraction of the business functions from the actual system on top of which it is implemented. However, even if today's SOA design principles are increasingly applied for new implementations, the existing content management applications tend to be of monolithic nature, so that a migration to a different backend system will be both, costly and time consuming.

IBM FileNet P8 takes this fact into account by delivering *content federation services* (CFS) as an integral part of its architecture. The main idea behind content federation is to understand the Content Engine as the master catalog

within the enterprise, which means that the Content Engine holds the metadata for *any* content object that is under management in the organization across *all* content management systems that are in use. The content federation layer in the Content Engine provides three important functions:

- ▶ It queries the third-party repository for new content, retrieves the metadata for that content, maps it according to the configuration, and creates a new object in the Content Engine with that metadata and the information of where the content itself is stored, which is a unique identifier for that content in the third-party repository.
- ▶ It queries the third-party repository for any updated content or metadata for objects that are already federated and updates the Content Engine information accordingly.
- ▶ When a client sends a request to retrieve the content to the Content Engine, it uses the appropriate CFS content connector to retrieve the content element from the third-party repository and delivers it to the client.

This approach is different from *search federation*, where at search time, the search is spawned across the configured third-party repositories and a combined result set is delivered to the client. In addition, an API is supplied that enables the retrieval of metadata and content from the third-party repository.

Apart from the federation capabilities, an enterprise catalog must provide strong capabilities to be useful and to handle federated metadata of content that is maintained in a third-party repository.

Duplicating the metadata information at the level of the Content Engine might, at a first glance, appear as waste because the metadata is actually stored twice: in the third-party system and in the Content Engine. Not necessarily all information must be copied to the Content Engine catalog, but it is possible to define only a subset of attributes to be transferred. As opposed to search federation, content federation has important benefits:

- ▶ Active content is seamlessly applied to the federated content.
- ▶ For any client, federated content is fully transparent, which means that there is no difference between content that is stored in the Content Engine or content that is federated into it.
- ▶ Federated content can therefore participate in any business process that is executed on the Process Engine and is automatically enabled for records management.
- ▶ Content Engine security can be applied to get a unified security model across all content that is under management.

- ▶ A unified taxonomy for content can be implemented on the level of the Content Engine, which makes it easier to find content throughout the enterprise.
- ▶ Content Engine and its sophisticated APIs can be used to implement new solutions that need to access content, which allows the implementation of a single source of truth of content within the enterprise. Together with the SOA capabilities of the IBM FileNet P8 Platform, this enables the customers to implement new agile applications that can easily adjust to changing conditions.
- ▶ Content federation allows a smooth transition if desired. Existing functionality can be implemented step-by-step on top of the IBM FileNet P8 APIs and can reuse the existing content in the third-party repository. No big bang migration is required because some applications can still work and change (or even add) content to the third-party repository, whereas other applications are already using the federated access through IBM FileNet P8. New content can then be created in the Content Engine and existing content can be moved behind the scene from the third-party repository to the Content Engine.

In this section, we illustrate how the IBM FileNet P8 architecture and its sophisticated capabilities help to solve common challenges that organizations face when they implement content management on an enterprise level.

7.1.4 Security and access control

Controlling the access to content is crucial for an ECM system for many reasons. As discussed in detail in Chapter 8, “Security” on page 187, it is mandatory to impose access control to the objects maintained in an ECM repository in order to ensure that the appropriate information can only be accessed by the correct audience. As we show in 7.3, “Content life cycle” on page 163, the authority to access content can change over its life cycle, which must be reflected accordingly. A typical example is records management, where a *legal hold* implies that all affected records must be locked, which means that their security must change.

A platform for content management on the enterprise level must meet the following requirements:

- ▶ Support fine grained access control on the level of *individual* objects.
- ▶ Quickly change the access granted to single objects and to a logical set of objects (for example the content objects which constitute a record).
- ▶ Imposed security not only has to be enforced for any access by any application using the APIs, in sensitive cases, it must also be possible to

provide auditing information about object level to prove that unauthorized attempts were blocked.

- ▶ The authorization and the access control must leverage the enterprise directory catalog so that it is not required to maintain the definition of security objects, such as users and groups, at the level of the ECM infrastructure.
- ▶ Security must be handled consistently across all modules of the ECM suite.

The IBM FileNet P8 architecture meets all of the above requirements. In Chapter 8, “Security” on page 187, we describe in detail how the different options can be utilized to implement sophisticated security and access control at an enterprise level and how features, such as object default instance security and marking sets, help to effectively handle security for content objects. The Content Engine and all custom created objects (for example, but not limited to content objects, content less objects, and folders) have access control information that determines which operations users can execute on these objects (and if they see them at all).

IBM FileNet P8 relies on a single interface to the directory catalog that is implemented by the Content Engine, and all other components, such as the Process Engine, leverage this information. Because all add-on products are implemented on the Content Engine and Process Engine APIs, it can be ensured that access control works consistently across all products. As an example, a process might have three documents attached and only one contains confidential information. By means of content security, different users working on this process can either see the confidential document or not depending on their role or group membership. Therefore, there is no need to set or hide attachments at the step level or when forwarding the process to certain users. If required, customers have the freedom to lock down the access to objects further on the level of individual applications.

7.1.5 Object classes and inheritance

Managing content for a large variety of users and use cases across an organization requires providing support to structure the various object types that can be stored in the repository. The Content Engine offers the concept of class hierarchies where subclasses can be derived from predefined foundation object classes.

Figure 7-1 on page 149 shows a document class hierarchy sample.

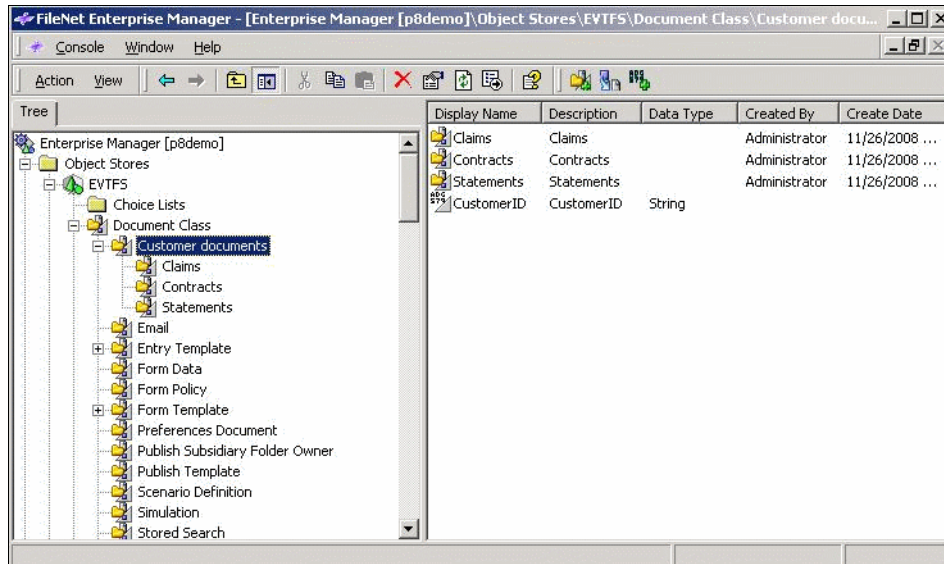


Figure 7-1 Document class hierarchy sample

As in object-oriented programming, the derived child classes share properties with their parent classes, and on the level of the subclass properties can be added. Access control is one critical building block for an ECM system. In the Content Engine, the security configuration is also passed from a parent class to its children but can be overridden on the child level. Changes to the definition at the parent level can be propagated automatically to the children, as required.

All foundation classes in the Content Engine support auditing, which means that the Content Engine maintains a protocol on events that can be configured to be monitored. We discuss this feature in more detail in 7.2, “Content event processing” on page 154.

Object class hierarchies enable consistency across the enterprise, and it also provides the flexibility that is needed to address requirements for content metadata for individual applications or departments. We now briefly describe the different foundation classes. For detailed information about designing a solution using the Content Manager class hierarchies, refer to Chapter 5 of *IBM FileNet Content Manager Implementation: Best Practices and Recommendations*, SG24-7547.

Objects with content: Document class

Objects with content are stored in the class *Document* or one of its subclasses. Objects that are instantiated in this hierarchy store the metadata and the information about the location of the associated content in the repository.

Instances of the *document class* hierarchy are *versionable*, which means that when a new version is created, the information about the content of the existing version is kept and can be reused for later purposes. It is not required that the content element is stored in the repository because the Content Engine supports the access to the content through federation, UNC and URL. In fact, the existence of content is not enforced. Objects can be created without having content associated.

When a new Object Store is initialized, several subclasses are automatically created, for example, IBM FileNet P8 applications, such as Workplace, use a subclass called *Preferences* to store user and site-related information as XML files in the Content Engine. Another example is the data documents that are created by eForms, which are also stored underneath a subclass of the *Document class*. With the enterprise perspective in mind, organizations will define all of the properties that documents need to share across the enterprise at the level of the foundation class and certain security information, such as read access to auditors.

Note: IBM FileNet P8 supports using content that is outside of the direct control of the Content Engine. Examples are federated content from other repositories and content that is accessed through a URL or UNC. The Content Federation Services (CFS) manages the federated content and takes care of any changes that are to be reflected in the Content Engine Catalog. Use care when you use content elements through a URL or UNC because this content can be changed without the Content Engines being aware that the changes occurred.

Objects in the Document class hierarchy can have zero or more content elements. Multiple content elements can either be added at once when the document is created or they can be added subsequently at a later point in time. Let us take an example: a document that contains multiple pages in TIFF format can be stored this way so that each page forms a single content element. This type of element is helpful especially when pages might need to be rearranged at a later time or when single pages must be displayed to the user at viewing time, so page caching is faster, which facilitates faster document viewing. To add further content elements, the document must be checked out, and after all content elements are added, it can be checked in again. Each version of the document can have a different number of content elements assigned. Further details of multi-content documents is in the ECM online help.

Contentless objects: Custom Object class

Even though ECM primarily addresses unstructured content, there are use cases where it is very helpful if the content management system can handle objects that only have metadata. The Content Engine uses the class *Custom Object* for

this purpose. Custom Object instances differ from *Document instances* in two important points. Custom Objects have no content and are not versionable. The latter feature is not a limitation: if versioning for objects without content is needed, the appropriate object class can be defined in the Document class hierarchy.

The Custom Object class hierarchy is used to model the *business objects* that are required for an application based on the ECM platform. There are several benefits to using Custom Objects instead of storing this information outside of the repository in a separate database:

- ▶ The same access control principles are imposed on Custom Objects that are imposed on content objects.
- ▶ Custom Objects support events and auditing (refer to 7.2, “Content event processing” on page 154) for more details.
- ▶ Custom Objects can participate in business processes or records management just as normal documents can do.
- ▶ Custom objects can be related easily to other objects that are maintained in the Content Engine utilizing Link Objects.

As an example, Audit Event information that the Content Engine maintains is stored as Custom Objects and related to the corresponding object instance that it belongs to, for example a specific document. Records that the IBM FileNet Records Manager defines are also modelled as Custom Objects, exploiting the features that the Content Engine provides to store the link between the record object and the elements constituting it.

Folders: Folder class

The *Folder* class hierarchy allows modelling folder structures that users are very familiar with when working with documents because they are accustomed to folders from their work with any file system.

A folder in the Content Engine shares many properties of a folder in a file system in that it is a container into which other objects (including other folders) can be filed and that it is access control enforced. It is possible to determine which security entities (users or groups) are allowed to file objects into a folder or are allowed to create subfolders. An object can be filed into any number of folders in the Content Engine and it is easily possible to obtain a list of all of the folders that an object is filed into.

Most importantly, the concept of active content also applies to any folder instance in the Content Engine, which means that events for the folder can be generated by the Content Engine, for example if a new document is filed into it. Besides, folders have properties and the folder class hierarchy can be used to

inherit the properties from parents to children. Folder properties allow efficient searches for folders compared to traditional browsing access only.

Folders are commonly used to structure content elements in some context, for example as a *case folder* that contains documents that belong to a certain case or a customer file folder structure that imitates the file plan that might have existed for paper-based documents.

Additionally, documents can inherit their security from folders through the SecurityFolder property. We explain this concept in more detail in Chapter 8, “Security” on page 187.

Link objects and compound documents

In the real world, objects are often related to each other. These relationships help us to structure and efficiently manage objects in our daily life. The same applies to objects that are stored in a content management repository. We already mentioned folders as a way to express a relationship and hierarchy between different objects that somehow belong to each other.

The Content Engine provides another powerful tool to express object relationships by providing the *Link Objects* class hierarchy. A Link Object has a *head* and a *tail* property of type *object* that determines the two objects that are related to each other. There is a 1:1 relationship, which means that if one object relates to N other objects, N Link Objects must be created to express this condition. Link Objects have properties and security, which means that different users might or might not see specific links. The sub classing of Link Objects is typically used to restrict the object classes that are accepted as head or tail for a certain Link Object class.

In addition the Content Engine provides a compound document framework. Compound documents are essential for ECM because there are many use cases, such as technical documentation or engineering, where a complex relationship between different objects must be expressed and maintained when the content is authored. In contrast to Link Objects, using the compound document framework you can also express complete hierarchies of relations, which is done based on *Component Relationship Objects*, which can be subclassed if required. Component Relationship Objects have a parent and a child, and any child can again be a parent for one or many other relations. The Content Engine provides efficient methods to explore the hierarchy of a compound document that is built on Component Relationship Objects. The compound document framework can also be used to maintain a document and its renditions into different formats or translations into different languages, for example.

There are important differences between Link Objects and Component Relationship Objects to express relationships:

- ▶ Link Objects have their own security, whereas Component Relationship Objects inherit security from the head object.
- ▶ Link Objects always point to a fixed version. The tail of a Component Relationship Object can be configured to point to either a fixed version, the latest version of the corresponding version series, or even to the latest version that has a certain label value.

These flexible options allow the compound document framework to express complex inter-object relationships that are found in areas, such as technical documentation.

Darwin Information Typing Architecture

Darwin Information Typing Architecture (DITA) is an XML Document Type Definition (XML DTD) that is an emerging standard that describes an architecture that was designed to organize and present technical content. DITA is designed to support authoring, producing, and delivering technical information. This type of content typically consists of various interrelated parts that can be authored independently. However, for managing DITA content it is crucial to provide extensive support to query for objects in the DITA structure because it is important to ease the reuse of DITA components.

The support for DITA takes advantage of the flexible Content Engine metadata model and extends it accordingly. The Content Engine provides a base document class for DITA content, which shares a minimal set of properties that are required for any piece of content in a DITA structure. Subclasses can be derived, which contain additional metadata, as needed. Another base class is used to define the hierarchy of the various content elements within the complete DITA document compound.

The metadata model for DITA uses the Component Relationship Objects described earlier to model the interrelationship of the various content objects that form the complete DITA document. The Component Relationship Object model allows efficient queries to explore the structure of the compound.

Note: The Compound Document Framework and the support for DITA are considered building blocks that allow customers and partners to implement applications that leverage these features. They are not exposed to users on the level of the Web application level (Workplace/WorkplaceXT).

7.1.6 Processes

Although we are discussing Enterprise *Content* Management and not Enterprise *Process* Management, there is no doubt that content will deliver its real value, namely the information it transports, always in the context of a business process. Refer to 7.4, “Business processes” on page 166 for a detailed discussion about content-centric business process management.

Not a single organization stores and manages content just to do it. The rationale behind content management is the fact that the content might need to be accessed at a certain time in the future. This request for the content is always triggered in a context of a process, be it one of the main business processes in the value chain of a company, such as access to an insurance policy when a claim is filed, or be it in a supporting process, such as approving invoices in accounts payable.

We do not mean that ECM will always require automating the business processes, but the largest value and benefit can be gained from content management that is tightly integrated into the business processes where individuals need information stored in the content management system for making decisions. Automating these processes is the first step towards optimizing the processes and thereby maintaining or obtaining competitive advantage. For this reason, the IBM FileNet P8 Platform utilizes the Process Engine as one of the base pillars in its unique architecture, which is the logical result of more than 20 years of experience that FileNet had in the area of workflow and business process management. We discuss the relationship between content and processes in more detail in 7.4, “Business processes” on page 166.

7.2 Content event processing

Content that is stored and used in a content management system is exposed to events, for example the request to display the content that exists for the simple case of imaging, the creation of a new version, or an update to an existing content object. In this section, we explain how the unique concept of active content helps to simplify the management of content in the context of ECM.

7.2.1 Active content

One of the unique differentiators of the IBM FileNet P8 architecture is the concept of *active content*. This principle is dedicated to the fact that managed content is always subject to events. In many use cases, depending on certain

incidents, specific actions must be taken. The most popular example is the creation of a document that launches a defined business process for further processing.

The IBM FileNet Content Manager implements *active content* by viewing a publish-subscribe schema. The Content Engine has the notion of *events*, such as the creation of a new object or the change of metadata for an existing object. Which *action* to take when this event occurs can be configured. For each event, multiple actions can be defined and for each of them it is possible to define filter conditions that further narrow down whether the action is taken. Filter conditions can be based upon all kinds of metadata that is related to the object that fired the event. Actions can be either scripts that are executed in the context of the Content Engine or Workflow being launched on the Process Engine, which we discuss in more detail in the following sections.

We previously mentioned that active content is not limited to objects that have content. Other Content Engine object classes, such as Folders, Custom Objects, or Link Objects also fire events that can be linked to configurable actions. Therefore, changes to business objects that are defined in the Content Engine can also directly interact with business processes on the Process Engine through active content. This way, not only the content itself, but also other information that is directly related to the content in the context of a business process can be maintained and handled in a uniform way, which is exactly one of the main benefits a platform delivers compared to an implementation based on discreet systems.

Active content can be viewed as a database trigger that is made accessible on a very user-friendly level and can be *configured* easily through the Web applications or the Enterprise Manager. The fact that using active content the events and the resulting actions can be configured rather than being coded enables the IBM FileNet P8 Platform to deliver a flexible process control and enables organizations to rapidly adjust to changing conditions.

7.2.2 System events

The Content Engine allows the firing off of events for certain incidents, which can be subscribed to and handled by a configurable event handler. This ability highlights the event-driven architecture of the IBM FileNet P8 Platform. The Content Engine allows the creation of an event when an important state is changed for a managed object. Table 7-2 on page 156 lists the system events that can be subscribed to.

Table 7-2 List of system events for the Content Engine

Event	Description	Subscribable Class or Object
Creation	Triggers when an instance of a class is created or saved or a reservation object is created (CheckOut).	Document, Folder, Custom Object
Deletion	Triggers when an object is deleted from an Object Store.	Document, Folder, Custom Object
Update	Triggers when the properties of an object are changed.	Document, Folder, Custom Object
Update Security	Triggers when the access control information for an object is changed.	Document, Folder, Custom Object
Change State	Triggers when the document life cycle state for a document is changed	Document
Change Class	Triggers when the class of an object is changed	Document, Folder, Custom Object
CheckIn	Triggers when a document is checked in	Document
CheckOut	Triggers when a document is checked out	Document
Cancel Checkout	Triggers when a checkout is cancelled for a document	Document
Classify Complete	Triggers when the classification for a document completes	Document
Promote Version	Triggers when a document is promoted to a new major version. Only available for Document classes that have versioning enabled	Document
Demote Version	Triggers when a document is demoted to a minor version. Only available for Document classes that have versioning enabled	Document
File	Triggers when an object is filed to a folder (including the creation of a subfolder)	Folder
Unfile	Triggers when an object is unfiled to a folder (including the deletion of a subfolder)	Folder
Freeze	Triggers when the Freeze method is called for a document	Document

Event	Description	Subscribable Class or Object
Lock	Triggers when an object is locked	Document, Folder, Custom Object
Unlock	Triggers when an object is unlocked	Document, Folder, Custom Object

The system events cover a large variety of object-related state changes. Additional *custom events* can be defined, and the subscription to these events and the configuration of the corresponding actions occurs in exactly the same way as for system events.

Auditing

Auditing refers to log information that is managed by the Content Engine for each object that has been configured to use this feature. When auditing is enabled for an object class, this means that each instance of this class, and by default its subclasses, is subject to auditing. The events that are subsequently captured and stored in the Content Engine for that object are configurable. Based on the object-oriented design, the audit configuration can be overwritten on the level of an individual instance or on the level of a subclass.

Auditing utilizes the Content Engines event model to capture the events. In addition to the system events that have been listed above, auditing allows you to catch some additional events that are not subject to subscription. For example, on the level of a document, the retrieval of content can be audited. For other objects (including documents), the retrieval of the property information can also be tracked. The audit configuration allows storing both, successful and denied attempts for the events, which is extremely helpful to document compliance for example.

All audit log entries will contain information about the event, the name of the user who performed the action, the date and the class and unique ID of the associated object and the result (*success* or *failure*). In addition, audit entries for some events will contain additional information like the properties that have been changed, or the text of an executed query, for example.

As the configuration of the Object Store is stored as objects in the Object Store database, auditing is also available for these system objects, which allows you to track who applied changes to configuration. For a complete list of auditable events and classes, refer to the IBM FileNet P8 ECM online help.

Each audit log entry is stored as a Custom Object in the corresponding Object Store database, which allows effectively querying the audit information. As

additional information must be captured and stored, a performance penalty must be calculated if auditing is activated. This can easily be done by using the Scout sizing tool as the creation of the audit log entries can be modeled in Scout accordingly as they are normal Content Engine operations.

Note: All events which can be subscribed to are also available for auditing.

The Content Engine provides functions exposed via Enterprise Manager to manage the event information stored, like export to XML and deletion of all audit information for a given user or a given object.

The auditing capabilities described in this section focus on the content objects managed by the Content Engine. They are also available for a system which does not (yet) use the Process Engine actively. Additionally, auditing information can be gathered on the level of the Process Engine and on the application level itself. Refer to 7.4.4, “Auditing and monitoring” on page 174 regarding details on auditing for business processes.

Which sources should be utilized to build the auditing heavily depends on the custom requirements for the audit trail, for instance, compliance might require that some audit information will be collected at the Content Engine level, whereas business related auditing can be implemented using audit capabilities of the Process Engine or at the application level.

7.2.3 Custom events

As we mentioned in the previous section, the Content Engine provides various system events that can be used to activate managed content and for auditing purposes. However, in complex situations the system events that are provided (out of the box) might not provide all of the required functionality. Using the flexible IBM FileNet P8 architecture you can extend the event model by adding *custom events*.

The Content Engine itself does not generate custom events; instead by a custom application through calling a RaiseEvent method for the corresponding object. This approach is beneficial because after the event is raised it is treated like a system event, which means that the corresponding event action (and filter conditions) can be configured using the Enterprise Manager.

Let us consider the situation, where a workflow is launched when a custom event occurs. Of course, it is possible to implement the workflow launch and passing parameters from the object to the workflow directly in the application itself (assuming, that the developer has the knowledge of the Process Engine API).

Using the custom event instead, you can configure which workflow is executed and which parameters are passed.

Changes to this configuration can be applied by the administrators without changing a single line of code. The changes themselves are subject to auditing, so they can be tracked if required. As a result, using a custom event you can enforce a common paradigm for the linkage between events and the corresponding actions and add flexibility to the overall solution.

7.2.4 Custom event actions

So far, we discussed that using the Content Engine's event-driven architecture you can raise system events and custom events and that the administrator can configure components that subscribe to these events to take the appropriate action. In this section, we describe the two interfaces that the Content Engine provides to allow custom components to perform event actions.

Interacting with external systems using event scripts

The first option to implement an interaction with an external system is by utilizing *Content Engine event components*. The main concept behind a custom event component is to provide a Java class that implements the desired action to be taken. This Java class must implement an `EventActionHandler` interface and can be made available in two ways:

- ▶ Upload the Java class file to the Content Engine server and provide the Java class path. For an environment consisting of multiple Content Engine servers, the module must be located on all servers at the same location.
- ▶ Define a *code module* object in the Content Engine that contains the jar or class file.

When defining the custom event action using the Enterprise Manager, the Java class is configured and optionally the code module object. After the custom event action is created, it can be used like any existing event action when creating a new subscription.

Synchronous and asynchronous execution

Subscriptions that use the Content Engine event components can be configured to be executed synchronously or asynchronously. Use the synchronous mode only if it is really required to execute the event action in the same thread as the originating activity to be able to roll it back in case the event action fails. A validation check for metadata fields when creating an object is one example, where in the case that the verification fails, the transaction of adding the content is subject to a rollback. The downside of synchronous execution is that the subscription processor on the Content Engine is locked until the event action

completes; therefore, only use it for custom event actions that can be completed very quickly.

Custom event actions do not necessarily require you to interact with an external system. One use case might be to check for the presence or status of a particular document or folder when the user tries to promote the current version of another document and to deny this action if certain criteria are not met, such as the folder does not exist or the status of the other document is not as expected, which is an example where synchronous execution can be used if the time to determine whether the required objects in the Content Engine exist is sufficiently short.

Note: We do not recommend implementing the interaction with an external system based on a synchronous custom event action because it cannot be guaranteed that the connection to this system is always active and that its response will always be immediate. As a result, a failure in immediate response by the external system eventually prevents the Content Engine from working because the event processor keeps the pending transactions open.

Asynchronous actions are executed in a separate thread, which means that the subscription processor can immediately continue to process another action and will not be locked until the event action returns. Therefore this is the preferred method to implement an interaction with an external system.

Note: Asynchronous action events are queued and executed in a separate thread. If multiple Content Engine servers exist in an IBM FileNet P8 domain, any server in this domain can be processing the request, not only the one that launched the event.

Interacting with business processes

Launching a process on the Process Engine is the second option that allows interaction with external systems based on a Content Engine event. The concept behind this approach is that either the process can communicate with the external system or that the process interacts with another active process that must be notified that the event occurred.

Communication with an external system can be facilitated using various methods, such as calling XML Web services or using a Component Manager step that performs operations on that system. We discuss business process management as part of ECM in more detail in 7.4, “Business processes” on page 166.

Workflow subscriptions can be defined and configured using either Workplace/WorkplaceXT or the Enterprise Manager. A workflow subscription

calls a defined version of a workflow. It does not automatically launch the latest version for this workflow, which allows transferring and pre-testing a new version of a workflow definition but still maintaining the production workflow subscription using the old workflow definition. If it is required to always launch the latest version of a workflow by a subscription, use a launcher workflow that starts the most current release of the business workflow. On the level of the workflow subscription, mapping properties for the object that launched the workflow to workflow fields can be configured and additional filter conditions that determine if the workflow will really be launched (for example launch a workflow only if a major version of a document is added).

Workflow subscriptions: Workflow subscriptions can be defined for objects of the Document, Folder and Custom Object class types.

7.2.5 Classification and taxonomies

Whenever an object is added to a content management system it is classified in a certain way. This classification is usually performed manually by setting appropriate metadata information, such as a document class or a folder into which the object will be filed. IBM FileNet Capture ADR can be used as an automatic classification because it can extract information from the scanned documents and use this information as metadata.

The term *taxonomy* is frequently used in relation to classification in IT systems and describes the list of entities or values that the metadata is allowed to have. In many cases, taxonomies are hierarchically organized, comparable to trees in a file system with folders and subfolders or the Document Class in the Content Engine.

The definition of a taxonomy is often derived manually by analysis of a limited corpus of objects, for example content management of documents. A good taxonomy allows the users to easily select the appropriate value for metadata information and also speeds up the search for documents because the taxonomy must avoid ambiguous metadata information.

The IBM FileNet P8 Platform provides support for classification in two ways, which we describe in the next sections.

Classification framework

The Content Engine provides an extensible framework that enables incoming documents of specified content types to be automatically assigned to a target document class and setting selected properties of that target class based on values that are found in the incoming document.

When a new document is created, a flag determines whether automatic classification is executed or not. If classification is enabled for the document, the Content Engine executes the following steps:

- ▶ The classification is performed asynchronously by queuing a classification request (with the reference to the document). The Content Engine ensures that the classification requests are properly queued and that the status for the classification is set to “pending” for the document
- ▶ Within a transaction:
 - The Document is handed to the Classification Manager, which determines the content type for the document based on its MIME type
 - The Classification Manager checks which Classifier must be invoked for the content type and passes control to the Classifier
 - The Classifier extracts the information from the document, updates the metadata accordingly, and passes back a status
 - The Classification Manager evaluates the status, sets the document’s classification status accordingly, and deletes the request from the queue

The Content Engine ships with one default auto classification module for XML documents.

Based on the interface definition provided, custom Classifiers can be implemented to support the automatic classification for other MIME types. By providing this framework, using the IBM FileNet P8 architecture you can change the configuration and the implementation details for the automatic classification without the changing any code for custom applications, which makes use of automatic classification for content.

IBM Classification Module

The IBM Classification Module (ICM) is a product that focuses on the categorization of content. Based on a given taxonomy, it performs an analysis of the content and proposes the taxonomy entry with the best fit, for instance a document class and a folder. ICM can be used to automatically assign metadata, such as the document class or a folder, in the Content Engine when the document is added.

ICM can also return a list of potential fits accompanied with the corresponding probabilities. This is a typical use case if a defined confidence level for an automatic classification is missed, and a manual classification must be performed. ICM *learns* whenever a user classifies or reclassifies a document, which ensures that a document with similar content is appropriately categorized in the future.

ICM also provides a *Taxonomy Proposer*, which is a tool that supports the creating a taxonomy based on the analysis of a reference corpus, which is a set of documents.

ICM can be integrated into the automated archival process that the IBM Content Collector manages, which enforces the usage of a common taxonomy and therefore a common classification for important metadata, such as the document class. Classification by ICM can also be launched as a post processing operation after the content is added to the Content Engine by employing active content.

Because ICM can manage various taxonomies, it is possible to use different taxonomies for different applications or departments across the enterprise, for example, one in R&D and another for HR or Finance.

7.3 Content life cycle

Content-based objects, such as documents that are frequently subject to a life cycle that spans from the initial version of this document up to a final release, might need to be kept for a defined retention period. Even for the simpler case of storing images of scanned documents that are processed in a workflow, there might be a change in the documents life cycle after the processing completes, and the document must be stored until its final disposition date. The Content Engine supports the definition of document life cycle and life cycle events that drive the document life cycle, which we discuss in more detail in this section.

7.3.1 Document life cycle

In the Content Engine a document life cycle consists of these objects:

- ▶ A *life cycle policy*, which defines the stages of the life cycle for this document
- ▶ The *life cycle event*, which fires when the document changes from one stage to another
- ▶ The *life cycle action*, which is linked to the event and can be used to perform custom operations

Life cycle policy

A life cycle policy can be inherited by a document through a definition at the level of the corresponding document class, or it can be assigned to the document at creation time. As described earlier, using the Content Engine you can override the inherited life cycle policy to a different one, if required.

Assigning a life cycle policy: Changing or assigning a life cycle policy can only be done when the document is created and cannot be changed at a later point in time.

A life cycle policy can contain permission information, which is automatically applied to the document when the life cycle state changes. The *changeState* method for the document must be called to change the state for the document, which can promote the document to the next state, demote it to the previous state, reset to the initial state, or set the document into an exception state that temporarily blocks the document for further transitions until the exception state is cleared.

Refer to 8.4, “Setting security across the enterprise” on page 205 for more details about how life cycle policies and security policies can be used to manipulate the permission control for objects in the Content Engine.

Life cycle action

Life cycle actions can be compared to custom event actions. They allow you to extend the capabilities of the Content Engine by assigning a Java class that is called to handle the life cycle event.

Note: Life cycle actions are always executed asynchronously. It is best practice to use a queuing mechanism to decouple the life cycle action and the actual execution of a custom operation in the case where this operation might run longer than the transaction time-out interval that is set on the J2EE server.

If the life cycle action and the actual operation that are executed were decoupled, the life cycle can be demoted in case the operation failed.

7.3.2 Life cycle and content storage

In some cases it is desirable that the life cycle of a document has consequences on the location where the document is stored, for example, a contract document that is created by iterating through several negotiation cycles, each represented by a new version of the document. After the content is finalized and approved from both sides, the latest version of the document, or a scanned image with the signatures on it added as the latest version, it might be desired to store this version on a fixed content device that implements WORM storage.

Even though the default life cycle actions of the Content Engine do not support changing the storage location for a content object, this is a good example of how the Content Engine event actions can be used to extend the capabilities. To

achieve the desired goal, implement either a custom life cycle action or a custom event action that uses the Content Engines moveContent method to move the latest content version to a fixed storage area. You can easily implement this operation based on the Content Engines APIs and provided as a code module for the custom action. Example 7-1 shows the code snippet for the moveContent method to change a documents storage area.

Example 7-1 Code snippet to move content storage area

```
private void moveContent(Document doc, String targetAreaName,
StorageArea sourceArea, String sourceAreaName, StorageArea targetArea)
{
    if ((targetArea != null)) {
        if (targetAreaName.equals(sourceAreaName) == false) {
            System.out.println("  Moving Content of document '"
                + doc.get_Id()
                + "' from StorageArea '"
                + sourceArea.get_DisplayName()
                + "' to StorageArea '"
                + targetArea.get_DisplayName()
                + "'");
            doc.moveContent(targetArea);
            doc.save(RefreshMode.NO_REFRESH);
        }
        else {
            System.out.println("  Content of document '"
                + doc.get_Id()
                + "' already stored in StorageArea '"
                + targetAreaName
                + "'");
        }
    }
}
```

If required, this concept can be extended to implement content aware storage, where the storage location for content is shifted based on the current state of the content within a workflow or its life cycle, for example from expensive to cheap storage or to a fixed storage area. This is another example that demonstrates the flexibility of the IBM FileNet P8 architecture because it allows an implementation that is superior over the capabilities of other ECM systems where the storage subsystem itself decides to move content to different storage locations because this is done only based on fixed retention times that were defined.

7.4 Business processes

Content that is maintained in a content management system is almost always in the context of a business process, which is due to the nature of the content being stored because it contains business related information.

Organizations that introduce a content management solution with IBM FileNet P8 do not necessarily also need to introduce business process management at the same time. Instead, the BPM capabilities of the IBM FileNet P8 Platform can be leveraged step-by-step. Using the flexibility of the IBM FileNet P8 architecture you can start with simple process support, for example for approval processes and to extend the reach of BPM to more and more business processes and business units.

We do not go into the details of BPM in this section, but we want to explain which business processes are an integral part of ECM and why process optimization, which requires BPM, is an important aspect.

Refer to Table 7-1 on page 142 for a definition of terms that we use throughout this chapter.

7.4.1 Content-centric BPM

Content-centric or content-aware BPM focuses on processes that are either driven by content objects, such as documents or heavily interact with content elements, either being created during the process or being used for the decision making steps in the process. Content-centric BPM differs from transaction-centric BPM, which primarily shifts data between different systems to support changes across heterogeneous environments. Refer to the *IBM FileNet Business Process Manager Reviewer's Guide*, REDP-4433, for more details about the different areas of BPM.

The most important features of the Process Engine for content-centric BPM are:

- ▶ The notion of an *attachment* data type that stores references to attached content objects. For a process definition, multiple fields for attachments or attachment arrays can be defined to act as a container for dedicated content objects that are linked to a process, such as an application form, the scanned image of an ID, and the contract document.
- ▶ The capacity for users to interactively attach content objects to a process with a direct integration into the ECM repository.
- ▶ The ability to modify content objects in background process steps.

Content-centric processes, in almost any case, involve human interaction at certain steps and the key concept is to convert data (commonly from different sources) into the information that is needed by the process workers to make their decision or perform their task. It is about *making better decisions faster by having the right information at the right time*. Content, especially of unstructured nature, such as e-mails, letters, or other documents, often build a critical part of this information.

For some businesses, content-centric BPM is one segment of their value chain, for example, financial organizations need to process a large number of customer applications for their products, such as credit cards or loans. The application forms are, in many cases, still based on paper or they are made available as an electronic form. In both cases, the content must be stored and managed in a content management system to ensure compliance and records management. The process of handling the application includes interaction with external systems, such as a core banking application, which can be integrated, as described in 7.4.2, “Complex interactions with external systems” on page 170.

Another example for a content-centric process is approving accounts payable invoices. Although it is only a supporting process for most organizations, the number of invoices that are handled by larger enterprises and the optimization of this process, for instance, to save cash discounts by releasing payments within a given time frame, is an important factor that allows you to save a significant amount of money. The approval and booking process in accounts payable involves the invoice itself, received either in paper or electronic form, and related information, in almost any case managed in an ERP system. Content-centric BPM can benefit this process in many ways:

- ▶ Storing the invoice in a content management repository enforces access control and retention management
- ▶ A Capture solution can be used to extract data from the invoice. It ranges from general data, such as a vendor name or number up to line item data
- ▶ The extracted data can be used to identify the corresponding record in the ERP system and to pre-fill the user interface or even perform prebooking steps automatically
- ▶ The corresponding approver can be selected automatically (or manually) and the invoice can be routed to that person for approval
- ▶ The BPM engine can monitor the SLA and perform escalations if required

The advantage of using the IBM FileNet P8 to implement content-centric BPM is the flexibility that the platform delivers to model business processes and their interaction with content. This advantage fosters rapid adjustments to the solution to change business requirements.

Access to content from the business process

Content-centric processes must provide access to content objects that are stored in the repository and they must support the creation of new content in the repository as part of the business process. The options that are available to accomplish this are:

- ▶ Default HTML step processors
- ▶ FileNet eForms and Lotus Forms
- ▶ Business Process Framework
- ▶ API-based step processors
- ▶ Web 2.0 user interface components

Default HTML step processors

The IBM FileNet P8 Web clients Workplace and WorkplaceXT, both allow the user to access his or her personal inbox and any other public work object queue, based on the permissions granted. When a user opens a process instance (or work object), it is represented in a step processor, which is an HTML page. This step processor allows the user to view the attached documents (assuming that the user has the appropriate permissions) and add new content to the process instance, if the attachment field is configured for writing access.

Process Designer is a tool that you can use to design your workflow process and configure the data and attachment fields of the work object to be shown in which step processor and what access is allowed (read, write, or read/write).

FileNet eForms and Lotus Forms

In many use cases, business processes are driven by forms or a form-like representation of data, which is desirable because the users are familiar with it because paper-based processes used similar forms in the past. To address this need, the IBM FileNet P8 Platform offers eForms. Electronic forms (eForms) are an HTML representation of a form that was designed using the forms designer tool. Therefore, eForms can be displayed in any Web browser that IBM FileNet P8 supports without needing to install additional software on the client PC.

Because eForms are built on the IBM FileNet P8 Platform, they integrate tightly with the Content and Process Engines. An eForm consists of a *form template* document, which stores the layout of the form, and information about lookups, verifications, calculations, and JavaScript extensions. When data is entered to the form, the *form document policy* determines which document class (and optionally folder) in the Content Engine to use to store the form data in an XML representation, the *form data document*. When form data is changed and saved, a new version of the form data document is created.

Additionally, eForms can be used for user interfaces in processes. The *workflow form policy* configures how form data fields and workflow data fields are mapped

into each other and which page of the form (for multi page forms) to use at which step in the process. eForms can have documents attached to them and users can add documents or change attached documents if they have the appropriate permission.

Using the Forms Integration Framework you can use Lotus forms in the same way that you use FileNet eForms.

Business Process Framework

The Business Process Framework (BPF) focuses on case management use cases and provides a highly configurable user interface for this purpose. Like eForms, BPF tightly integrates with the Content and Process Engine and allows integrated access to content objects that are part of the case. Based on the permission of the user, new content elements can be added to the case or existing documents can be replaced by a new version. BPF also enables users to add files from the local file system to a case that automatically adds the content to the repository as well.

API-based step processors

If the out-of-the-box capabilities delivered by the products previously described are not sufficient, custom step processors can be implemented to act as the user interface to the process, for example, external applications can be integrated seamlessly into the user interface or a thick client can be implemented. A custom step processor will use the Content and Process Engine API to access process data and content information and information from external data sources or applications and represent them the way that all requirements are met. As the logic of business process is defined on the level of the Process Engine workflow definition, it can be altered to meet changing requirements without changing the user interface.

API-based step processors can utilize all of the features that we described earlier in this chapter, such as creating custom life cycle events, and they can use all of the features of the IBM FileNet P8 Platform, such as auditing or records management (if installed).

Web 2.0 user interface components

IBM FileNet P8 Version 4.5 introduces a new user interface to business processes that is based on Web 2.0 technology. It delivers the benefit of allowing you to change the user interfaces rapidly using a widget-based approach. Consider a widget as an area in the browser window. A designer can define the user interface by placing the widgets in a design application at the desired position inside of the browser window.

The widgets follow a common specification, which allows them to communicate to each other and to exchange data between each other, for example:

- ▶ A process data field widget that has the ZIP code as one of its data fields to pass this information to a map widget, which then automatically displays an area map for this ZIP code
- ▶ A customer number from the work object data widget that is passed to a master data widget that collects and displays additional information for this customer, such as name or address

This approach makes user interfaces to processes even more flexible and allows the reuse of widgets across different applications.

7.4.2 Complex interactions with external systems

We previously discussed, in “Interacting with business processes” on page 160, that a business process allows complex interactions with external systems. We now provide more detail regarding this approach in the remainder of this section.

Component Manager

The Component Manager provides a way to call custom java classes from a step on a workflow map, allowing configurable passing data from the process instance to the java component and retrieving the result back into the process instance. The IBM FileNet P8 Platform ships with one component (CE Operations), which allows interaction with the FileNet Content Engine to modify content objects. Starting with release IBM FileNet P8 4.5, the component is extended to also access content in the IBM Content Manager V8 repository.

Custom components can perform virtually any operation and especially facilitate the interaction with external systems, for instance, if an architecture heavily relies on Java Messaging Queues (JMS), such a component can be used to read and write messages from JMS queues. If a java class is already implemented to execute specific operations on an external system, such as performing an update to a master database or updating a record in a host application, this java class can easily be re-used by adding the jar file (and any dependant Java libraries) to the Component Manager's Java class path and configure which methods of the Java class, at the level of the Process Engine, are made available at process steps and which parameters need to be provided for each method call.

The Component Manager uses the flexibility and the open architecture of the IBM FileNet P8 Platform to allow a direct integration with external systems at the level of a workflow step.

Process orchestration

Complex business processes involve operations on many different systems throughout their execution. In many cases, parts of the process might already be implemented in a given workflow environment. The main idea is to use industry standards to link these systems together. Instead of re-implementing the complete process on a new BPM system, only the basic process flow is modeled on this BPM system and this process is *orchestrated* by calling functionality from an existing system or even starting a process on another workflow system and waiting until the result is returned so that the main process flow continues.

The rationale behind process orchestration is to better leverage the existing assets and capabilities of the systems that participate in the business process, for example, the Content-centric part of a process might only be an exception route of the main process, which itself is running on IBM WebSphere Process Server because it heavily involves the transformation of business objects that are defined for different systems into each other. Staying in WebSphere Process Server requires implementing a human task with integration into the IBM FileNet P8 Content Engine repository to display a document that the user needs to make a decision, how to resolve the process exception. Instead, the exception process in FileNet BPM can be launched from the WebSphere Process Server engine, passing the reference to the document. The Content-centric exception process can leverage the tight integration into the Content Engine to display the document (and possibly other related content objects) and the FileNet BPM process sends the result back to the WebSphere Process Server process, which continues its execution.

The Process Engine supports process orchestration by utilizing the Business Process Execution Language for Web Services (WS-BPEL). This standard defines how Web service calls can be used to perform interoperation between business process runtime engines, which serves two use cases:

- ▶ In the first use case, a process running on the Process Engine can call a Web service on an external system to perform an operation (*invoke* step) on this system, and it can wait to retrieve the answer that the system passes back (*reply* step) as a Web service message. The Process Engine provides rich XML parsing capabilities to form the Web service messages and to extract data from received messages.
- ▶ In the second use case, processes on the Process Engine can be exposed as Web services to be called by external systems. Thus processes that execute on the Process Engine can also orchestrate business processes that are driven on other BPM systems.

Custom work performers

Custom work performers are the most flexible option to enhance the capabilities of the Process Engine or to integrate processes with external systems. A custom

work performer can be considered as a background process that regularly polls a queue on the Process Engine for new work. Custom work performers use the Process Engine API to query the queue, lock a work object, read data, perform the configured operation, write data back to the work object (optionally), and dispatch it to the next step.

Compared to the solution of implementing a java class and using the Component Manager, a custom work performer can be implemented as an executable, a service, or a demon process that allows, for example, easier access to system resources, compared to the java class, which is executed within the Java Runtime Environment (JRE™) that the Component Manager runs in. The advantage of utilizing the Component Manager is that the logic that is required to launch the component at system startup and query the queue for new work is already built into the Component Manager, whereas it must be implemented for a custom work performer.

Service-oriented architectures

Service-oriented architectures (SOA) are considered to be an important design pattern to build applications that allow businesses to quickly adopt to new business needs due to changing market trends, regulations, and so forth. One foundational principle of a SOA is to build reusable components, the *services*, which are interconnected to deliver the required functionality. The services are meant to be carved to encapsulate a business-related functionality (such as *retrieve customer file*) as opposed to IT-related functions that are often used in today's architectures (such as *retrieve document*). The abstraction of the business function (the service) from its actual IT implementation also allows the changing of the implementation of isolated services without re-implementing large parts of the application, which is built on top of the services. In fact, if the interface to the service does not change, the application does not even need to be touched.

Consider process orchestration for Content-centric BPM as one layer in a SOA, as shown in Figure 7-2 on page 173. On the bottom level, we see the data layer on top of which the services are implemented. The process layer bonds together the business process and uses process orchestration to call the individual services. The BPM engine, which drives this layer, not only ensures proper routing but also implements important features, such as timers, to track the execution and progress for the service calls and exception handling. The process workers, which are sketched on the top layer, interact with the processes by the user interfaces that the BPM system provides.

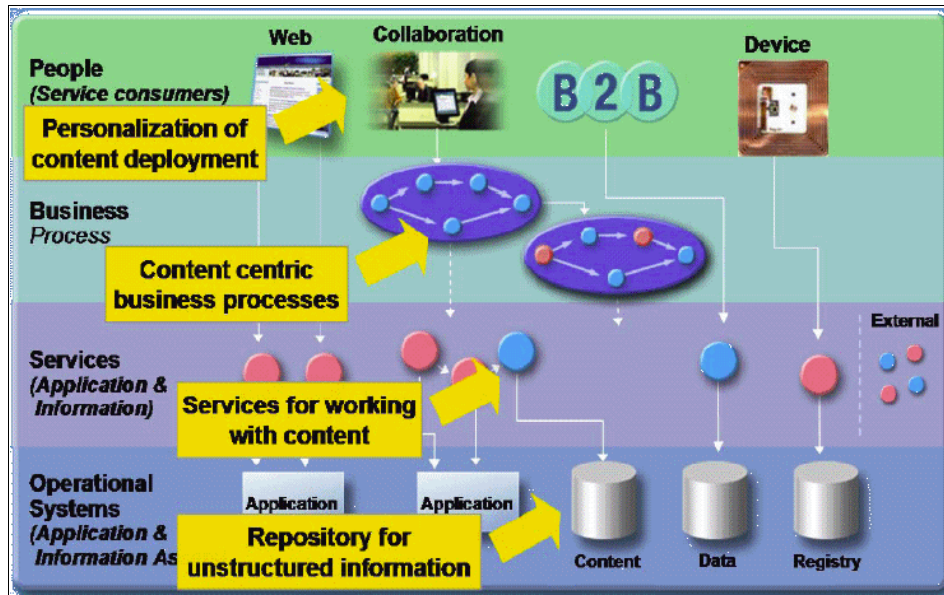


Figure 7-2 ECM and SOA

The IBM FileNet P8 Platform can either participate in applications that are designed on a SOA or the Process Engine itself can drive Content-centric business processes using process orchestration to interact with external systems and other BPM systems.

Although we mentioned SOA in conjunction with business processes and the Process Engine, services that only use Content Engine functionality can also be defined. These services can then be exposed to be used by an SOA based on the communication mechanism that is defined for the architecture.

7.4.3 Business Rules Engines

Conditional routing is one of the common requirements for any BPM system because process instances must follow different paths, depending on defined rules, for instance, an incoming loan request might need to be routed to a certain team or even a team member based on information, such as the ZIP code, the amount, and name of the applicant. Another example is a claim process that is routed to a junior adjuster if the claim amount is below a given value and must be processed by a senior adjuster above that amount.

The Process Engine supports the specification of such rules at the level of the process definition, but this approach has one important impact. The rule for a process cannot be changed after it is instantiated, which is required for certain

compliance scenarios. Changing the rule requires creating a new version of the process definition and transferring it to the Process Engine so that it gets applied to any new process instances.

Alternatively, for less complex rules like the second example, an option is to store the threshold value in an external database and read it at decision time from the process, which you can do out-of-the-box using the provided integration of the Process Engine to call database stored procedures directly from a process map. This way, it is possible to update the threshold value without the changing the process definition. The downside of this approach is that another tool is required to maintain the values in the database. In many cases, the business units can modify those values without routing it through the IT department. A Business Rules Engine (BRE) is one flexible way to handle this problem. The business rules are stored centrally in the BRE, which provides interfaces so that arbitrary applications can then use the rules. In addition, many BREs support the definition of the rules in a user friendly business vocabulary, which makes it much easier for the business units to define and maintain the rules in the BRE. Another important benefit of using a BRE is that it allows enforcement of consistent rules across applications because the rules are defined only in one place and re-used from different applications. This process matches the idea of re-usability.

The Process Engine supports using external business rules in a process by calling BREs from a process at runtime. The rule name and the parameters to pass are configured at the process definition level. At execution time, the Process Engine calls the BRE using a Rules Engine Framework. The BRE evaluates the rule and sends the result back to the Process Engine. Based on the result provided, the Process Engine dispatches the process instances into the appropriate route. The Rules Engine Framework plugs in BREs from different vendors, such as ILOG, Corticon, or FairIsac.

By using a BRE, the processes that are executed on the Process Engine gain the flexibility that conditional routing can be performed based on business rules that are centrally stored and maintained by business units. BREs are very helpful in Content-centric processes because they can be used to define the routing of documents to inbaskets on group or individual levels, externally, to the process definition. This ability improves the agility of business processes because they can immediately be adjusted to changing conditions in the way automated decisions are made in the process.

7.4.4 Auditing and monitoring

In this section, we cover auditing and monitoring processes from a business perspective, rather than from an IT or systems management point-of-view.

Auditing

The term of auditing a business process is frequently used in connection to compliance. The goal is to demonstrate that the process follows given rules or regulations, which is usually done by investigating the history of the steps that were executed and why decisions were made.

Apart from the option that a custom application can log the appropriate information for auditing purposes, the Process Engine and the Content Engine can provide audit logs on the level of individual process instances or content objects. Custom information can be written to these logs by properly configuring the IBM FileNet P8 Platform.

Process Engine

The Process Engine can log custom information for a process on the level of the process definition, which means that the process map is enriched with system steps that are writing custom entries to the event log database.

Business Process Framework

When using the Business Process Framework (BPF), every case has a dedicated audit trail that is maintained on the case level in the Content Engine and can be directly accessed through the BPF Web application. It is also possible to trigger custom log entries for the audit trail at the process map level.

Content Engine

The Content Engine can log audit information can for individual content objects based on either system or custom events. We previously discussed the underlying concept in “Auditing” on page 157. If the audit information that the Content Engine logs is supposed to be used in the context of business processes, it must be ensured during the design phase that information about the business process that triggered the change on the content object must be passed to the content object. This is related to the chain that a change, which was applied to the content object, triggers the event that causes the audit entry being generated.

Monitoring business activity

Business activity monitoring addresses the supervision processes in progress on the Process Engine. IBM has two offerings to use for this purpose in conjunction with the IBM FileNet P8 Platform:

- ▶ IBM FileNet Business Activity Monitor (BAM)
- ▶ IBM WebSphere Business Monitor (WBM)

Ultimately, both options serve the same needs because they can be used to present live information from the BPM back end system and from other data

sources, in a real-time fashion. The data is gathered from the configured sources, aggregated to represent key information, such as service level agreements (SLA) and typically other key performance indicators (KPI) and thresholds. To represent this information to the business, monitoring dashboards, which represent the information by gauges and other intuitive graphical elements, are typically used.

Figure 7-3 shows a sample monitoring dashboard that BAM created.

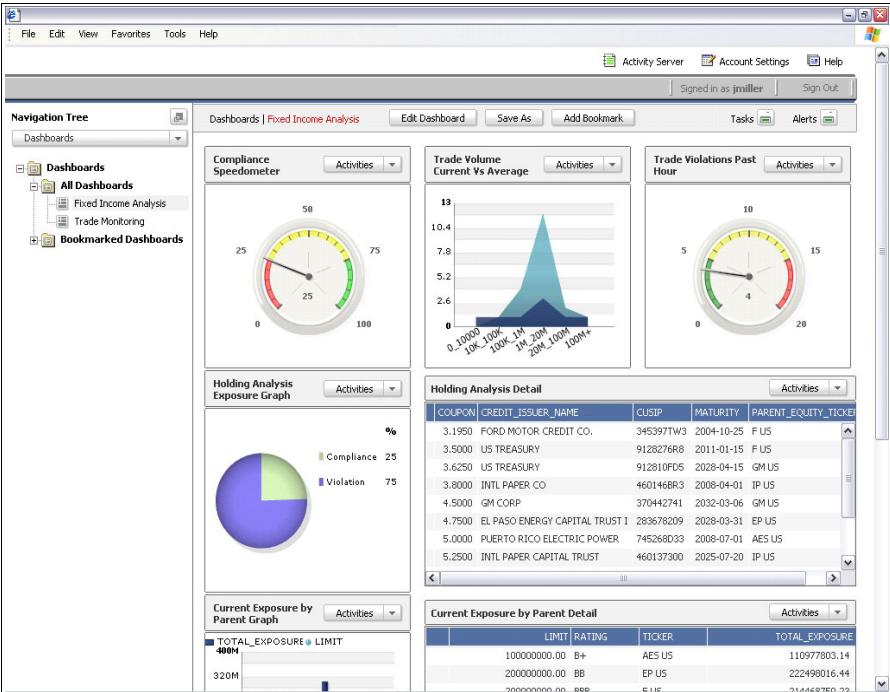


Figure 7-3 Monitoring dashboard visualized by BAM

The gauges and figures are updated frequently and represent the most current status of the business process that is being monitored, which allows the business managers to detect bottlenecks or problems in the process and to take immediate action.

Additionally, using BAM you can define rules for monitored threshold values and add actions that BAM automatically executes when the rule is violated, for example, you can specify that for a loan application process, the SLA for a sequence of steps is four hours.

Whenever a process takes longer than that, immediate action could be configured, such as:

- ▶ Alert the supervisor
- ▶ Provide listing of the affected processes
- ▶ Execute corrective actions, for example by making a Web service call to remedy the problem)

On the level of the workflow definition, you can implement other actions, for instance, setting a timer and alerting the supervisor when the timer expires. However, BAM makes it more convenient to collect the data from various sources and aggregate and extract the KPI information. Additionally, dashboards enables business analysts to spot trends or critical changes just by looking at the graphs and gauges.

BAM supports the inclusion of data from other sources that are being imported and used to derive the status for the KPIs and SLAs under control. This supports allows including information that are gathered from other BPM systems. This enables to monitor business processes which span across BPM systems.

Common Base Events

The Common Base Events (CBE) specification is a standard XML-based format for business events, system events, and performance information. CBEs are intended to be consumed by monitoring and analysis tools.

CBEs can also be used to implement monitoring on the level of business processes, for instance, WebSphere Business Monitor (WBM) provides a dashboard that is very similar to BAM. WBM can also be used to monitor the Process Engine, which is another option to watch processes that span across BPM systems because WBM also supports WebSphere Process Server. To enable the monitoring of Process Engine by WBM, the Process Engine Event Logs are directly accessed by a CBE adaptor and converted to a CBE format, which WBM can then further process. Figure 7-4 on page 178 shows the architecture of this configuration.

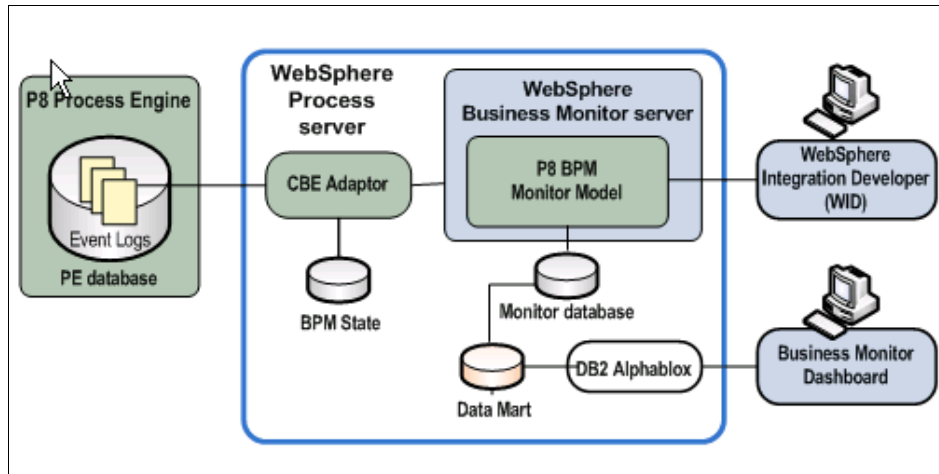


Figure 7-4 Common Base Events for Process Engine

The CBE Adaptor reads the configured event logs from the Process Engine event log database and transforms the events into the CBE format. The CBE information is sent to the WBM Monitor Server. Based on the monitoring model for IBM FileNet P8 BPM, which was configured on WBM, the corresponding information is passed to the monitoring database. Data is frequently transferred from the monitoring database to the Datamart. The DB2 Alphablox analysis technology is exploited to extract the KPI information from the Datamart and display it on the dashboard.

In summary, different options exist to implement business activity monitoring for the Process Engine. Which one is preferable depends on the customer requirements and architectural constraints because BAM requires MSSQL Server for the Process Analyzer.

7.4.5 Analysis and optimization

As opposed to business activity monitoring, business analysis and optimization focuses on data that is gathered over a time in the past. The time span being analyzed might be days, weeks, or even years to analyze trends and draw conclusions.

IBM FileNet P8 Platform supplies the Process Analyzer tool to analyze business processes. Process Analyzer leverages Microsoft MS® SQL Analyzer to supply the data in a format that can be quickly explored and drilled down by users. The Process Simulator tool is also available to perform what-if simulations of the process model to discover bottlenecks in the process execution. Analysis and

simulation aim for continuous improvement in the quality of the business processes. For this optimization, it is necessary that the process is executed on a BPM system to efficiently collect the matrix.

Data from the Process Engine event logs is fed on a schedule into a datamart database. This datamart stores the data in a special representation (snowflake/star schema) as opposed to the flat schema that is used by the event logs' tables, for example. In a second step, the OLAP cubes are calculated from the current datamart information. There are basic OLAP cubes, which are provided during the installation of the PE, and customers can define new cubes if required. The configuration of the cubes is stored in the Microsoft SQL Analysis server. Figure 7-5 outlines how data from the Process Engine event logs becomes available in the OLAP data cubes for further investigation.

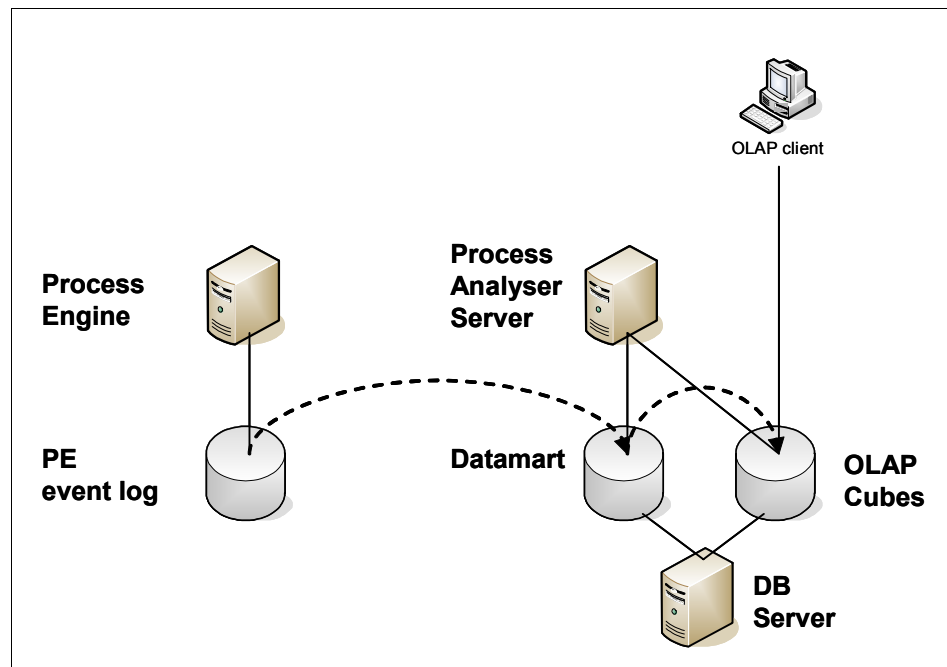


Figure 7-5 FileNet Process Analyzer data flow

The OLAP cubes can be inspected using an OLAP client, for example, IBM Cognos or Microsoft Excel. Process Analyzer installs a number of predefined Microsoft Excel spreadsheets that use the base OLAP cubes to generate reports for information, such as process execution time, step completion time, queue load, and much more. Use the reports to perform a *slice and dice* analysis, which means that the data viewed is narrowed down further to see details, for example, a report might show the number of completed transactions over a time period, and then you can check, how the transaction value affected the processing time

or how the transactions are distributed over different regions and if there are differences in the average transaction value for the regions. Assuming that these details are gathered from the event logs and are stored in the datamart, the above analysis scenarios can be performed by just a few clicks.

Based on the results of the analysis, the Process Simulator (PS) can be used to determine which changes to the process definition must be applied to eliminate given inefficiencies or bottlenecks. To do so, you can reuse the process definition, alter it, and load it into the PS. For each simulation, a scenario is defined that consists of the process model, arrival times, work shifts for manual processing steps, and (optionally) costs. Arrival times can be defined manually or derived from the production Process Engine. The scenario is versioned in the Content Engine and handed over to the PS. The PS uses statistical methods for the arrival times and simulates the flow of the process instances on the workflow map. The PS provides basic measures for the simulated process, such as process execution times and costs. In case a deeper analysis of the simulation is desired, a PA instance can be attached to the PS. In this configuration, it is possible to further analyze the simulation results with the PA, as previously described.

The optimization of Content-centric processes is an important building block in the architecture of an ECM platform. Continuous optimization of the business processes is a critical asset for organizations to keep their competitive advantage even in changing business conditions. Leveraging the flexibility and the support of a wide range of standards of IBM FileNet P8 Platform allows enterprises to roll out solutions for their Content-centric processes that can easily be adjusted to changes in the way they perform their business, thus ensuring a low TCO compared to applications that are individually developed.

7.5 Records management

In this section, we use the IBM FileNet Records Manager to highlight how this add-on product leverages the IBM FileNet P8 Platform capabilities to address records management. It shows that compliance requirements link very well to IBM FileNet P8 Platform features, such as events, audits, permissions/access control, and processes integration for content. IBM FileNet Records Manager provides an API for implementing custom solutions. It also serve as an example of how the IBM FileNet P8 system can be further extended to satisfy special requirements.

7.5.1 Basic requirements

In this section, we provide a brief summary of requirements that are typically encountered in records management and compliance use cases.

A record is any type of content stating results achieved, pertaining to, and providing evidence of activities performed. A record has the following characteristics:

- ▶ Fixed content in either physical paper format or in electronic format
- ▶ Evidence of a transaction, activity, or fact that has legal or business value
- ▶ Specific retention period based on company policy and regulatory rules
- ▶ Owned by the company, enterprise, or government

Records management involves at least the support for the following operations:

- ▶ Defining a file plan to store records
- ▶ Identifying the information that needs to be declared as record
- ▶ Categorizing the records
- ▶ Retaining records for a specific period of time
- ▶ Destroying records when an organization is no longer obliged to retain them
- ▶ Preserving an audit trail of all activities related to the records

Two key factors in records management are:

- ▶ Records preservation: Ensure that records are maintained and accessible until the appropriate retention period elapses
- ▶ Records destruction: Ensure that records are destroyed after the required retention period ends

Be prepared for the dynamic nature of records management, for example, documents might need to be declared as records based in certain incidents, such as when litigation occurs or that existing records must be kept longer for similar reasons.

For a detailed introduction of records management, refer to the first chapter of *Understanding IBM FileNet Records Manager*, SG24-7623. Also refer to this book for more in depth description of IBM FileNet Records Manager.

7.5.2 Product features and leveraged IBM FileNet P8 Platform capabilities

IBM FileNet Records Manager uses several features of the IBM FileNet P8 Platform from both Content Engine and Process Engine. Some of these features are working behind the scenes while others are exposed to the end user through the IBM FileNet Records Manager Web application extensions.

File plan management

IBM FileNet Records Manager uses the concept of individual Object Stores to separate the *document objects* from the actual *record objects*. Document objects are stored in a records-enabled Object Store (ROS). Record objects are stored in a dedicated File Plan Object Store (FPOS). Record-enabling an object store is an easy task that only adds a few metadata objects to the content objects in the ROS to maintain the link to the associated record objects.

The FPOS stores the records objects. Although they might be stored in the same Object Store as the ROS, in most cases they are stored in a separate, dedicated Object Store. Figure 7-6 illustrates the separation of document and record objects. Separating the FPOS allows sharing a common records schema across different ROS without adding the records and file plan-related metadata to each ROS. Additionally, this separation enables federated records management for third-party repositories and the management of physical (paper based) records.

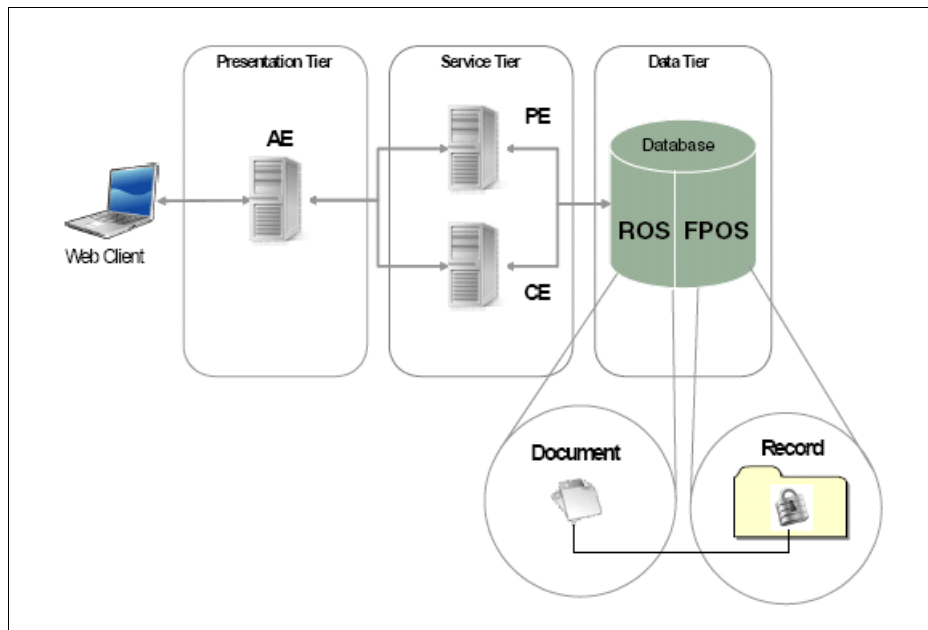


Figure 7-6 Separation of document objects and record objects

IBM FileNet Records Manager makes use of the object class hierarchies to define the business objects needed, for example:

- ▶ File plan objects
- ▶ Record containers
- ▶ Record categories:
 - Record folders
 - Record volumes
 - Record objects

IBM FileNet Records Manager makes extensive use of the Content Engine's security features, for instance, IBM FileNet Records Manager leverages markings and security proxy objects to effectively change the security for a whole record, which might consist of over hundreds of documents or even more. A rapid change of the security of multiple objects is a critical requirement for records declaration and for operations, such as a records lock or the disposal. Additionally, a security change must propagate to the children if it was changed on a higher level of the hierarchy.

Flexible record declaration and classification

IBM FileNet Records management solutions are as diverse as the potential customer base for an ECM system. There are large differences in the way and the volume of document ingestion. Some organizations might be satisfied with manual document entry, such as using Workplace and a manual declaration, because the business users who add the documents have the skill to decide if a document is a record and if so, of which type. A workflow might be present, for instance, when an approval is required by another person before a record can be declared. Other enterprises might have a large number of documents that are added and that might require declaration as a record, for example, e-mail. In this case, it might be desirable to lift the burden of manual record declaration from the users' shoulders.

IBM FileNet P8 supports many different use cases by providing a wide spectrum of records declaration options. Records can be declared interactively by users, and rules-based automatically, which is known as *ZeroClick* records declaration. Using the Component Manager, it is easily possible to declare a record from a workflow step, thus integrating records management and BPM. Records can be declared, locked, and unlocked directly from business processes.

Alternatively, records can also be declared as part of a custom event action, which is possible because the IBM FileNet Records Manager provides a Java API that can be called from a custom event component.

When a document is declared as a record, the record must also be classified, which basically means that the record must be placed into a certain position in the file plan. The document metadata is an important source for classifying a new

record. Additionally, consider using the IBM Classification module to classify new content and to use the result for a decision, if a document needs to be declared as a record.

Record life cycle management and active compliance

A record usually has a defined time span, the retention time, until it becomes subject to disposition. The term disposition does not automatically equal record destruction. Other options are a review or a transfer of the record to another system, such as a paper based archive.

Which steps need to be taken when a records retention time has expired, heavily depends on the individual organization and is in most cases described in a process rather than a single step. Therefore, IBM FileNet Records Manager tightly integrates with BPM to allow the definition of various workflows that can be triggered depending on the record life cycle, for instance a review of records prior to their destruction can be easily implemented this way. Using Component Manager, it is possible to destroy a record from a workflow step.

Because the records management workflows are running on the Process Engine, there is a huge degree of freedom in how an organization can model their individual workflows to match closely with their records management strategy.

Note: The workflows that are launched on particular events of a record life cycle are executed on the Process Engine. Therefore they can easily be adjusted to match any organizations' individual requirements, for example, on records disposition.

Records hold

Retention periods for records are fixed, but in the event of a litigation, audit, or similar, pertinent records must not be destroyed, which is referred to as *records hold*. Record holds are dynamically placed on existing records as a response to certain business events.

IBM FileNet Records Manager can manage multiple holds simultaneously because large organizations might be subject to different, independent events that require a hold, such as multiple litigations or a litigation and an audit.

Federated records management

We already mentioned that IBM FileNet Records Manager is not limited to managing records for documents that are stored in a Content Manager Object Store. Instead, using Content Federation Services, 7.1.3, "Enterprise catalog and content federation" on page 145, it is possible to extend records management to

third-party repositories that lack a solid solution for this purpose. Even though the document objects in this case are not directly maintained by the Content Engine, all other features of IBM FileNet Records Manager, still apply, including integration into workflows.

Federated records management requires that IBM FileNet P8 can modify the security for content objects that are stored in the third-party repository. This is required because the document content must not be changed after it is declared as a record.

Record search and retrieval

Based on the rich metadata model of the Content Engine, you can easily search for different IBM FileNet Records Manager objects, such as file plan entities. By leveraging the search templates for the Content Engine, which can be considered as canned search definitions created by an administrator or a power user who can restrict the end user in the way they can search the system. Search templates are not only used to support and ease the process of searching for specific information, they also prevent users from executing searches that potentially slow down the system because they initiate full table scans on the level of the database.

Search templates are used in various contexts within the IBM FileNet Records Manager Web application, for example to manually find records that must be put on hold or to find records that are subject to disposition.

7.5.3 Platform extensions

IBM FileNet Records Manager not only provides an out-of-the-box (OOTB) Web application, which can be used or modified to implement an individual records management solution for an enterprise, but also provides a Java API that extends the capability of the core IBM FileNet P8 Platform in the context of records management requirements. Based on this API, customers can implement their own records management very effectively, even when the OOTB features do not fit their needs, for instance, an organization might have policy to use rich clients instead of Web applications. Refer to *Understanding IBM FileNet Records Manager*, SG24-7623 for information about IBM FileNet Records Manager Java API.

7.6 Summary

Throughout this chapter, we provided many examples of how organizations can benefit from the IBM FileNet P8 architecture when trying to establish content

management on an enterprise level. It is important to understand that content management over the past years was implemented at the departmental level only as the users felt that the capabilities that the vendors offered did not fulfill their needs. You can compare it to the market for structured content about 15 years ago, where the market was fragmented and most departments used their own database application with a local database engine because they felt that a centralized approach did not address their needs or was not adapted too slowly to changing conditions.

Today, the market situation is changed. Many organizations made a global decision for a certain database vendor; however, it is a good practice that departments still have a solution that exactly fits their needs, but they would use a database infrastructure centrally provided by IT to store the data. This centralization gains benefits, such as centralized backup and restore and the ability to maintain a single view on the truth for critical information, such as master data instead of having this information duplicated and unsynchronized across different databases across the different departments.

Using the IBM FileNet P8 architecture customers can make a similar platform decision for content management. Today any kind of business faces the requirement that it must be able to adjust to changing conditions. The flexibility of the IBM FileNet P8 Platform combined with the rich set of features and its open APIs allow customers to implement true ECM across all units of their organization.



Security

Each IBM FileNet P8 product has its own functionality, but they are all built on top of the IBM FileNet P8 Platform with Content Engine, Process Engine, and Application Engine, which we described in earlier chapters. The support for security around authentication and access control of processes and content of these products is provided by the core platform. In this chapter, we describe the security issues to consider in an enterprise environment, how IBM FileNet P8 addresses them, and how to manage security effectively in an IBM FileNet P8 environment.

We cover the following topics:

- ▶ 8.1, “Authentication and authorization” on page 188
- ▶ 8.2, “Securing IBM FileNet P8 core components” on page 192
- ▶ 8.3, “Access to information” on page 196
- ▶ 8.4, “Setting security across the enterprise” on page 205
- ▶ 8.5, “Security requirement changes with time” on page 219
- ▶ 8.6, “Content-level security” on page 227
- ▶ 8.7, “Network security” on page 229
- ▶ 8.8, “Reporting and auditing” on page 233
- ▶ 8.9, “A practical example: Re-insurance placement and litigation” on page 240
- ▶ 8.10, “Summary” on page 247

8.1 Authentication and authorization

In this section, we describe how the IBM FileNet P8 Platform interacts with other enterprise systems to perform authentication. We also discuss the different types of objects, services, and operations that can be protected using access control, and how authorization is carried out for those elements.

8.1.1 Authentication in IBM FileNet P8

IBM FileNet P8 must use a directory server to authenticate against and be able to look up user group membership information using Lightweight Directory Access Protocol (LDAP). Starting with version 4.0, IBM FileNet P8 uses the application server's built-in support for authentication to a central directory server. IBM FileNet P8 currently supports the following directory servers:

- ▶ Microsoft Active Directory 2003, 2008
- ▶ Microsoft Active Directory Application Mode (ADAM)
- ▶ IBM Tivoli Directory Server 6.0 and 6.1
- ▶ Sun Java™ Directory Server 5.2 SP3, 6.x (6.x support in IBM FileNet P8 4.5)
- ▶ Novell eDirectory 8.7.3, 8.8.x

A full list of the supported authentication providers is in the IBM FileNet P8 Hardware and Software Support Guide, which is available from the IBM Web site.

Typically, authentication is provided by a user logging in to one of the Web user interfaces, such as Workplace. Workplace takes this user name and password and wraps it into an object called a Java Authentication and Authorization Services (JAAS) context. This context is then authenticated by the application server and referenced when communicating with the Content and Process Engines using the Java API.

The application server intercepts the JAAS context when a call to the IBM FileNet P8 Platform is made. It looks to see which of its configured JAAS login modules work with the context and then use it to authenticate the user. After authentication is successful, the application server informs the Content Engine that the user is valid and authenticated.

Because of this approach, the only piece of information that the Content Engine uses from the JAAS context is the identifier of the user. Regardless of the JAAS module used, this identifier must be consumable by the Content Engine's configured LDAP user and group lookup filters. Typically this is accomplished by using the LDAP common name field, for example, the user name on the system.

The Content Engine does not use JAAS to get any information about group membership, roles, or permissions on objects. All group membership is looked up through LDAP by the Content Engine. All permissions are stored in the relevant Content Engine or Process Engine databases for that particular user's unique identifier. For an IBM FileNet P8 system that is linked to Active Directory, for example, the unique identifier is the user's SID, which means that the LDAP directory schema does not need any changes to it to support an IBM FileNet P8 solution. The only change that is required is in creating users and groups to manage security of the system.

Lookup: In IBM FileNet P8 4.0 and above, all authentication and group membership lookup is delegated to the Content Engine. The Process Engine no longer performs its own lookups directly to the Directory Server, which simplifies the configuration of Authentication and Single Sign-On.

Although the JAAS standard is designed to abstract authentication and authorization, it occurs at the cost of mapping constructs into a tight JAAS model. The Content Engine has very fine-grained support for access control, so it cannot use the Java security access control model. Many JAAS providers also do not support the more advanced security features of the most common directory servers. An example of this is Microsoft Active Directory's support for multiple forests with multiple domains. As such, IBM FileNet P8 performs its own group membership lookups and maintains its own access control lists internally.

This greater flexibility required the developers to write code specifically to talk to each directory server. As a result, IBM FileNet P8 supports a specific subset of the most prevalent directory servers that are available on the market.

8.1.2 Single Sign-On

The term Single Sign-On, or SSO, is often misused and generally implies that there is no manual sign in required in several scenarios. For clarity, we use the following definitions of methods of sign on:

- ▶ Traditional Authentication

User name and password-based sign on made manually by users.

- ▶ Single Sign-On

This is when a client authenticates to the first SSO-protected service that they use and does not have to log into any other. In other words, users sign in only once. Before a user request is handled, a third-party application, called an SSO provider, intercepts the request and authenticates the user. This user session is then asserted as valid to the application, so the user is not again

asked to supply their user name or password, which is the process when using SSO software, such as Tivoli Access Manager with WebSeal.

Typically, the same gate keeper SSO software is used across multiple Web interfaces and signing into one of these means the user does not need to sign into the others. Web application SSO protection systems, such as secure reverse proxy software like WebSeal, provide their own login page instead of using that of the underlying services.

A common mechanism for SSO is Kerberos, which is the underlying technology that is used in Windows Integrated Login, and uses the same method when passing Kerberos tokens to applications. For Windows Integrated Login, the user's machine retrieves the Kerberos token from the Directory Server during initial login and passes it directly to the application server to validate. In this situation, the application server itself acts as the intercepting third party. Instead of the application server presenting a login prompt, it asks for a token but does not prevent access to the underlying application if it is not present.

► Token passing

Token passing occurs between two Web-based applications that share a trusted relationship. Within Workplace, for example, a user might click a hyperlink that opens a Records Manager application. This URL also holds the user token so that Records Manager does not ask the user to log in again.

Notice that Single Sign-On requires third-party software to validate the user, which allows developers to abstract out any authentication code from the underlying application. Organizations can implement the same security access restrictions across all compliant applications, which is exactly what the Content Engine supports through JAAS.

The method used to assert credentials to a service where access is controlled by gate keeping SSO software can be one of two types. Under the first approach, some products can be configured to fake the login to the application, which allows the SSO software to look up the user name and password and passes that into the login box of the Web application being used. This logs in the user in the background and then performs the user's request. The protected application has no idea that the client has not typed in an actual user name or password. This method is very useful when adding SSO support to older applications that do not check for JAAS contexts.

The second method, and the method that WorkplaceXT uses, is to assert a JAAS context to the application server and have the application check for a valid context before forcing a manual login. This context can then be passed through the application stack to other JAAS-enabled software with which it communicates, such as the IBM FileNet P8 Content Engine and Process Engine.

Figure 8-1 shows a typical process for how Single Sign-On works with a third-party SSO provider.

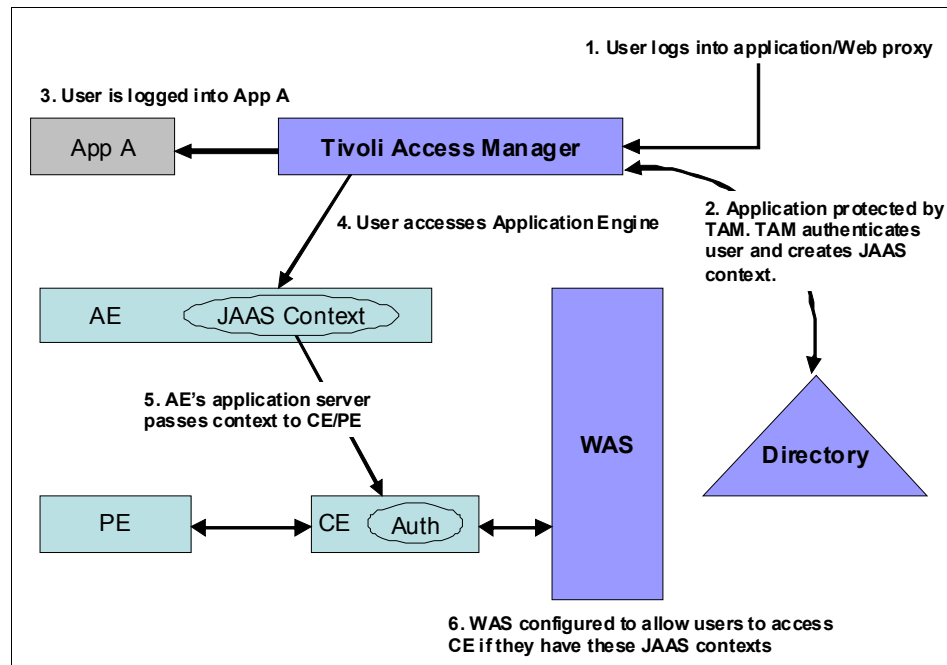


Figure 8-1 Typical Single sign-on (SSO) authentication behavior

Some application servers support token or SSO-based authentication themselves. An example of this is the WebSphere 6.1 support of the SPNEGO client handshake method and in particular the Kerberos token that can be retrieved using that method. This requirement is common, especially in organizations that widely use Windows Integrated Login for their employee-accessed applications.

Note: The Content Engine can be used with any JAAS context configured in the application server. WorkplaceXT, however, is only supported against the SSO products' JAAS contexts listed in Table 8-1 on page 192 and the out-of-the-box user name and password context.

There is a wide range of SSO authentication options that are available today. Unfortunately, they are typically tightly linked to the application server and directory server being used. Figure 8-2 on page 199 summarizes the supported SSO environments for IBM FileNet P8. A full list with precise IBM FileNet P8 versions that are supported for each configuration are in the latest IBM FileNet

P8 Hardware and Software Support Guide, which is available on the IBM Web site.

Table 8-1 Supported platforms

Product	Platform(s) supported
CA SSO / eTrust SiteMinder 6.0 SP3	WebSphere 6.0.2.13+ WebSphere 6.1.0.9+ WebLogic 8.1.5.x WebLogic 9.2 (IBM FileNet P8 4.5)
IBM Tivoli Access Manager with WebSeal 5.1.x	WebSphere 6.0.2.13+ WebSphere 6.1.0.9+
IBM Tivoli Access Manager with WebSeal 6.0.x	WebSphere 6.1.0.9+ WebSphere 7.0 (IBM FileNet P8 4.5)
Kerberos for Web services clients 2003	WebSphere 6.0.2.13+ WebSphere 6.1.0.9+ WebSphere 7.0 (IBM FileNet P8 4.5) WebLogic 8.1.5.x WebLogic 9.2.x WebLogic 10.x (IBM FileNet P8 4.5) JBoss 4.0.2
Kerberos for Web services clients 2008	WebLogic 10.x (IBM FileNet P8 4.5)

Kerberos for Web services: Kerberos for Web services clients is supported for the Content Engine Web services API but not for the Process Engine Web services API.

8.2 Securing IBM FileNet P8 core components

The IBM FileNet P8 Platform's core engines have much functionality that can be secured independently. In this section, we give a high-level summary of each component and the circumstances under which it makes sense to secure them. We then discuss these areas in more detail in the following sections in this chapter.

8.2.1 Content Engine

The Content Engine has the richest set of authorization access controls of any part of the IBM FileNet P8 Platform, which is necessary to satisfy a range of use case scenarios. When we think about security for the Content Engine, we

naturally think about access to document content and metadata but, in reality, access control goes far beyond that.

The Content Engine supports a wide range of permissions, some of which are applicable only to certain types of objects, for example, the *Major Versioning* or *View Content* permissions are only applicable to instances of the document class and its subclasses. The *View All Properties* permission on the other hand is applicable to documents, custom objects, folders, and many more objects in the Content Engine. We describe the most commonly used permissions in 8.3.1, “Document security” on page 196.

At the heart of all the methods of assigning permissions in the Content Engine is the concept of *Access Control Entries* (ACEs). An ACE entry links a permission to a user or group in the directory server. ACEs are contained within an *Access Control List* (ACL) or Permission List in IBM FileNet P8 terminology that is attached to a Content Engine object.

Permission Lists are assigned to an object and are not classed as objects in their own right, which means that you cannot assign the same permission list to multiple objects. The Content Engine does, however, check whether newly created permission lists are identical to ones that already exist within the system. If the new permission list is the same as one that already exists, the Content Engine only stores one copy. This function is transparent to applications that are built on top of the Content Engine and allows for very efficient caching of permissions.

The security objects that the Content Engine supports are Marking Sets, Security Policies, Document Life cycle Policies, Dynamic Security Inheritance Objects, and Default Instance Security descriptors, and we describe each of these in subsequent sections. An instance of a Content Engine object, such as a Document, maintains its own list of instance security, independent of the above security mechanisms. This is populated when an instance is created by the Default Instance Security settings of the document class or specified explicitly by the application that added the document.

WorkplaceXT, for example, has the concept of Document Entry Templates. There can be many types of *Entry Templates* for a single class of document that the user chooses. These templates can set default values for properties, a default filing location, and can specify security ACEs. They are application-level constructs that are independent of built-in descriptors that are used within the Content Engine.

Entry Templates are very useful in certain circumstances, for example, if a customer visits an organization's Web site to complete a registration form, they should not need to be concerned with its class or filing location. All of this can be handled behind the scenes by the Entry Template.

An interesting by-product of the highly granular security model in the Content Engine is that you can lock system administrators out from viewing the content using marking sets, but still allow them to effectively maintain the system. This feature is especially useful in organizations such as those in Financial Services, Health Care, or Defense, where the need to secure information separately from the infrastructure might be a requirement. As it is possible to lock such employees out of accessing Content Engine information, you can ensure that stringent security policies cannot be circumvented through administrative or maintenance access.

8.2.2 Process Engine

The Process Engine has a concept of *User Queues*, *Public Queues*, and *Component Queues*. A User queue restricts access to work items in that queue to only the specific user that is assigned in the workflow definition. A Public Queue, on the other hand, is accessible by multiple users and groups who all share a similar role within a workflow.

Although it is called a Public Queue, it is possible to limit access to specific users and groups and specify which of those can view work items or take action on them (for example, complete the step and move the work along in the process flow).

The following list contains IBM FileNet P8 terminology that is used in the context of workflows:

- ▶ Queue: A list of active work items grouped logically. A queue can contain several types of activities to be worked on by the same team.
- ▶ Step: Also called a task. An item of work to be completed by either a user or a background system.
- ▶ Process Map: Also called a workflow definition. An executable definition of steps, routing conditions, and fields to be carried out by the Process Engine.
- ▶ Process: Also called a workflow. A running instance of a Process Map with its own data values and audit history.
- ▶ Isolated Region: An area in the Process Engine database that contains all work items and queues for a particular application.
- ▶ Roster: A list of all work across all queues within an Isolated Region.
- ▶ Business Process Management Suite (BPMS): Distinct from simple document workflow because it includes tools for simulation and analysis of processes. A content-centric BPMS also supports process orchestration and other integration techniques in addition to tight integration with an Enterprise Content Management (ECM) system.

A Component Queue is used to interact with instances of a Java class to perform some system task, which could be to file a document in a particular location, set or fetch a document's properties, or to update document security. Because this component is a system process, it is possible to specify a particular JAAS stanza (configuration) to use and the user name and password to pass to the configured login modules, for example, login modules exist for user name and password authentication to the Process Engine, Content Engine's 3.5.x Java API, and Content Engine's IBM FileNet P8 4.x Java API.

It is also possible to construct a component that asserts its own custom JAAS context using standard Java code for cases where you might need to interact with other third-party systems. This action is dependent on how the component is coded by the developer and independent of IBM FileNet P8.

Certain IBM FileNet P8 add-on products extend the security that is available in the Public and User queues to restrict the list of work items that are displayed in the interface. Business Process Framework (BPF), for example, can be configured to specify an inbasket configuration that restricts a particular queue to a specified role, which can be further restricted by specifying that only a single step type is shown in this inbasket, or by constructing a queue filter to show only certain items based on the value of a process property. A good example of this is when there are junior and senior staff members with differing access authority. A Junior Approver queue filter can be created to restrict the list of pending credit approvals to those credit requests that are less than \$10,000. This is also a useful way to limit the number of work items being displayed for a particular queue, as it is possible to create more than one inbasket that lists a subset of the work items available to that user in the queue.

8.2.3 Workplace and WorkplaceXT access roles

The WorkplaceXT user interfaces extend the concept of user and group-based access control by restricting who can see and use certain functionality that is available in the interface to particular users and groups.

A WorkplaceXT Role is simply a list of users and groups, for example, an Authors role can be created to allow only certain people to see the “Add document” or “Check in” user interface Actions. This approach covers all Actions that are available from the right-click document and folder menus, and on the property pages for objects. Roles restrict who can see the main pages, such as Browse, My Workplace, Downloads, and Advanced Tools.

Roles are commonly used to restrict who can see advanced authoring tools, such as the Search Template Designer or Process Designer. There is even a special role, called P8BPMPProcessDesignerEx, that lists who can see the

Microsoft Office Visio® diagram import option within the Process Designer applet.

Business Process Framework has its own concept of roles. BPF can be configured to use LDAP group integration where the role name is the same as a group in the directory server. Another alternative is to link a BPF role directly to an existing Workplace role with the same name, which is useful if your users access both user interfaces because it is only necessary to specify the members of each role one time, rather than for both products. Another benefit is that users of BPF can have access to their queues removed from the Tasks interface of Workplace, which forces them to go through BPF to work on their tasks.

My Workplace acts as a mini portal where custom pages can be configured to show certain folders, search results, or actions. These pages can be restricted down to certain Workplace access roles. This restriction is particularly useful where an organization wants to deploy a configurable portal solution for ECM and BPM but does not want the overhead or cost that is associated with a full-blown enterprise portal solution.

8.3 Access to information

In this section, we describe the different methods that are available within the Content Engine for creating access control lists. This information is available in more detail in the IBM Redbooks publication, *IBM FileNet Content Manager*, SG24-7547-00, but it is of importance to the whole platform because all add-on products are built upon this security framework. In this section, we make specific reference to where each method is used within expansion products, and we show how overall security benefits from a platform architecture approach.

8.3.1 Document security

Each document maintains its own list of access control permissions, known as Access Control Entries (ACEs). When several of the ACEs are attached to one object, it is known as an Access Control List (ACL). These ACLs can be specified at the document instance level to specify who can access particular documents. These ACLs only make sense when they are attached to a document because they are not independent objects within the system. There are methods whereby ACLs can be reused across multiple objects, which we discuss later.

Everything within the Content Engine, from document class definitions to documents to saved searches and folders, can have its own ACL. The result of having an ACL on a document class is that it can be masked when a user checks in a document and is prompted to select its class, which is particularly useful in a

system that is used across departments, preventing the Legal team, for example, from checking in a Customer Request Document.

To assign individual access rights or permissions, simpler security levels can be used to group permissions together in the administration interface, as shown in Table 8-2.

Table 8-2 Permissions applicable to Document subclasses

Rights	Security levels						
	Full control	Minor versioning	Major versioning	Modify properties	View properties	Publish	View content (default)
View all properties	Y	Y	Y	Y	Y	Y	Y
Modify all properties	Y	Y	Y	Y			
Reserved 12*	Y						
Reserved 13*	Y						
View content	Y	Y	Y	Y		Y	Y
Link a document/Annotate	Y	Y	Y	Y		Y	
Publish**	Y					Y	
Create instance	Y	Y	Y	Y			
Change state	Y	Y	Y	Y			
Minor Versioning	Y	Y	Y				
Major Versioning	Y		Y				
Delete	Y						
Read permissions	Y	Y	Y	Y	Y	Y	Y
Modify permissions	Y						
Modify owner	Y						

Rights	Security levels						
	Full control	Minor versioning	Major versioning	Modify properties	View properties	Publish	View content (default)
Unlink document		Y	Y	Y		Y	
Create subfolder (inherit only)							

* Indicates deprecated permissions.

** Indicates that it is defined in Workplace to include Modify permissions.

Some of the rights in Table 8-3 are applicable to only some Content Engine objects. Table 8-3 is a list of which permissions are applicable to which objects.

Table 8-3 Object classes and permissions that affect access to them

Content Engine object class	Applicable permissions
Applicable to all	▶ View all properties ▶ Modify all properties ▶ Create instance ▶ Delete ▶ Read permissions ▶ Modify permissions ▶ Modify owner
Document	▶ View content ▶ Link a document/annotate ▶ Publish ▶ Change state ▶ Minor versioning ▶ Major versioning ▶ Unlink document
Folder	▶ Create subfolder ▶ File in folder/annotate ▶ Unfile from folder
Custom Object	Link/annotate

Assigning permissions: It is possible to assign permissions to a Content Engine object that is not applicable to that class. This action is implemented to allow the inheritance of permissions from one object to another. See “Dynamic security inheritance” on page 213.

8.3.2 Default instance security

Each class definition can specify the Default Instance Security to apply to a new document when it is not provided by the application that is built on top of the Content Engine. Figure 8-2 shows an ACL that is specified on the Class.

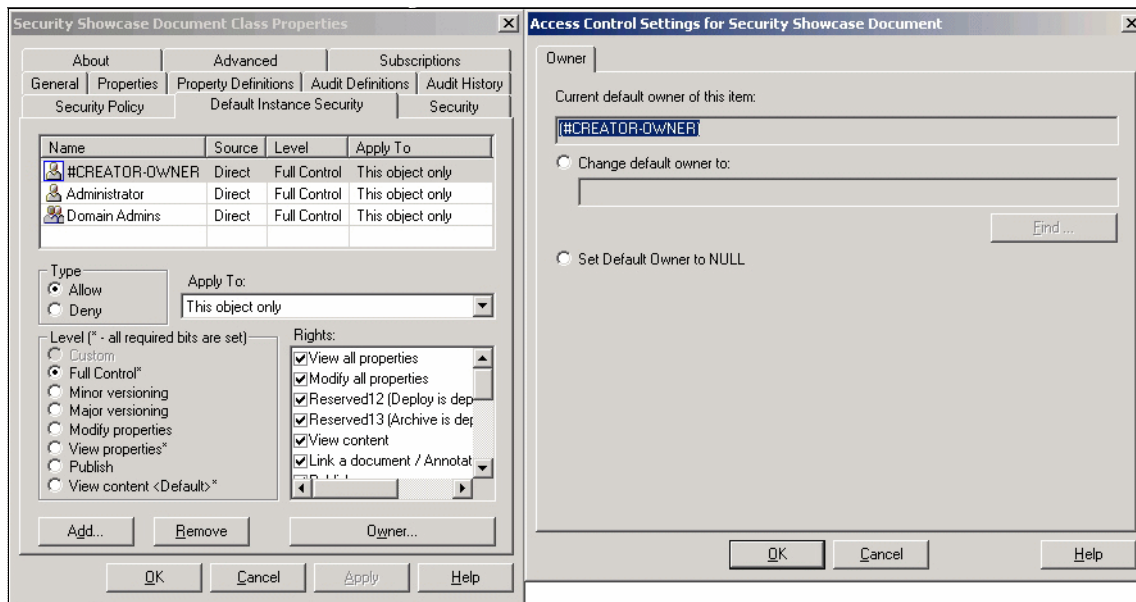


Figure 8-2 Default instance security and default owner settings on a document class

It is also possible within this interface to specify default instance security ACEs that apply to new instances of this class. If a subclass of this class is created, these ACEs are copied to the new class. However, updating a higher-level class does not automatically update subclasses.

Within these ACLs, two special entries can be used:

- ▶ The #CREATOR-OWNER allows rights to be assigned to whomever added the document (or whomever is specified in the Owner field at creation time)
- ▶ The #AUTHENTICATED-USERS entry can be used to refer to any user who is logged into the system.

The Owner of a document instance can also be specified at the class level. By default, this is left as #CREATOR-OWNER, meaning that the owner of the document is specified as the user who created it. This rule can be overridden, however, to point to another user or group on the system. The Overridden owner of the document can be useful when the Owner must always be assigned to a particular user or group of users. If you specify the #CREATOR-OWNER to have

particular permissions on a document instance, rather than on a class' default instance security, then these permissions have no effect. When the document is created, #CREATOR-OWNER is determined and that user is assigned rights on the new document instance. So, if user1 logged in and added a document, and the default Owner field is left as #CREATOR-OWNER, any rights assigned on the default instance security tab to #CREATOR-OWNER are assigned to the user1 on the new document instance. In other words, the #CREATOR-OWNER permission entry would not exist on the instance. It would instead show user1 as assigned to those permissions.

8.3.3 Object owner permissions

Folders, Custom Objects, and Documents all have an owner property, which can be blank or point to a particular user or group. Whoever is assigned as the owner receives the following rights on the object:

- ▶ Read permissions
- ▶ Modify permissions
- ▶ Modify owner

By virtue of having the modify owner right, the owner also receives the read all properties right on the object; otherwise, the owner cannot see the owner property and therefore cannot modify it. If a user or group is given the modify any owner right on an object store, they are also given read all properties and modify owner rights on every object in the object store.

Markings and their constraints, which we discuss in 8.4.1, “Marking sets” on page 205, are evaluated after the owner and object store permissions are assigned. As such, markings can be used to deny permissions that are granted to the owner of an object.

Note: A group can be assigned to the owner property, which is very useful where ownership of a document lies with a team rather than a user. Setting the owner to a group in this case gives anyone in that team the owner's rights. Over time, the group membership can be modified and no change to the owner field or access control list is required.

Note: The owner property is evaluated after direct, template, and inherited permission sources. Any denial of those levels for the rights conferred by being an owner are overridden, and the owner still gets those permissions on the object.

8.3.4 Security precedence and inheritance

Typically, an organization wants to ensure that only authorized users have access to any content in the system. It is quite rare for any user in an organization to have access to perform all operations on a document. There are instances where all users might have read permissions, such as for policy documents, but generally, all content is locked down to some extent. This lock down can be done explicitly for each and every document, but this is cumbersome and hard to manage.

There are often cases where it is useful for security to flow through various objects to a document. The classic example is in foldering, where the security on the document should reflect that of the containing folder. Another example might be a workgroup, such as an IBM Redbooks publication team, that specifies that a certain group of people have access to a set of documents. Perhaps the most classic example is security classification. These classification groups are created at an enterprise level and must be enforced to override or mask security permissions that are already on individual documents.

Before talking about how authorization is calculated, we discuss the precedence of how these various security sources affect the target object. The following list shows the highest priority security source first, followed by lower priority sources:

- ▶ Default or Direct Deny
- ▶ Default or Direct Allow
- ▶ Template Deny
- ▶ Template Allow
- ▶ Inherited Deny
- ▶ Inherited Allow

Permissions: A higher source of permission always overrides that of a lower source.

A source of default indicates that the security permission was assigned to the document through the default instance security mechanism that we described earlier. A direct permission is the same, except it is assigned to the document instance directly, rather than copied from the default instance security permission list.

Security Templates are assigned to a document based on the settings within its assigned Security Policy. These are permissions that are copied into a document's access control list (Permission List in IBM FileNet P8 terminology) when the document's state matches a corresponding state in the policy that has a security template configured. When a match exists, the permissions are copied to the document.

Security template assignment: A security template assignment only happens when a life cycle event occurs or an application explicitly assigns an application security template and not when the security is evaluated.

This behavior is similar to that of a document life cycle policy. The difference is that a document life cycle state change occurs when a custom application calls one of the methods to modify a life cycle on a document, not in response to a versioning action, which is the case for security policies. A particular document life cycle policy state might or might not be configured to apply security permissions. For more information about Document Life cycle Policies, see 8.4.3, “Document life cycle policies” on page 211.

Inheritance specifies that the security is carried forward from another object, which is different from the previous approaches because the ACEs are not copied into the document but are dynamically evaluated on-the-fly. This method is particularly useful from a management perspective because many objects can inherit security from the same source object. This method is used by the Security Folder property on a document and folder to indicate the object from which security settings are inherited. It is also used in property-based dynamic security inheritance that is described in “Dynamic security inheritance” on page 213.

8.3.5 How authorization is calculated

Extra security ACLs for a document, folder, or custom object can be inherited. Security is calculated in the following order with each rule taking precedence over those that follow it in the list:

1. Direct and Default permissions that are assigned to the document instances are evaluated. A permission is marked as default if it is applied from the document class' default instance security settings.
2. Template permissions are evaluated. A Template permission is assigned by a document life cycle state change or security template. See “Security policies” on page 209 and “Document life cycle policies” on page 211.
3. Inherited permissions are evaluated. These permissions can be from the SecurityFolder field or any Object Value Property that is defined with a Security Proxy Type of Inheritance. See “Dynamic security inheritance” on page 213:
 - a. If a Security Folder is specified, it is checked for any ACEs that are specified to apply to the selected choice of entities: *This object and immediate children* or *This object and all children*. A document can be filed in multiple locations but can have only a maximum of one Security Folder. This value is not set by the WorkplaceXT applications. If the

security folder is null (that is, not specified), no folder security is evaluated at any level.

- b. If this folder has its *Inherit from security parent folder* setting enabled, then this folder's parents, starting with the immediate parent, are checked for all ACEs that apply to *This object and all children*.
 - c. If all parent folders have *Inherit permissions from parent* enabled, then eventually the root folder's security is checked, which is the top level of potential security inheritance for folders. By default the root folder gives write permission to all users for all properties of any contained object in the object store. It is important to change this security on a production system.
4. The owner of the document is checked. If this is the user trying to perform the current action, this user is granted *read all properties*, *read permissions*, *modify permissions*, and *modify owner rights* on the document.
 5. The containing object store's ACL is checked. Some object store permissions affect documents, folders, and custom objects. These permissions are *write any owner* and *privileged write*. The first of these permissions allows the specified grantee to change the object owner. The second allows a grantee to modify certain properties on any object in the object store. These properties are creator, date created, last modifier, date last modified, and date checked in.
 6. If the document has a property whose value is from a marking set, the specified marking is evaluated. If the current user does not have the use right on the marking, then the security permissions specified in the constraint are removed (masked) from the computed permissions list. Thus if a user has modify content rights but is not granted a marking's use right, the constraint can be configured to remove this right.

In 8.9, “A practical example: Re-insurance placement and litigation” on page 240, we give a practical example using multiple sources of security, including inherited security, to illustrate how setting security at one level can override other security without causing any problems or gaps in the platform security landscape.

8.3.6 Authorization calculation example

Now we walk through an example for authorization, assuming that we need to control the view content right on a document instance. Table 8-4 on page 204 shows the security settings that affect this particular document instance and its view content permission. We do not discuss owner or object store granted permissions to keep the discussion simple. We assume, for our example, that none of the users are the document owner or have special rights from the object

store permission list. The underlined permissions are those that have the highest precedence in our example.

Table 8-4 View content permissions and their sources for our example document

Permission source	User A	User B	User C	User D
Direct Deny		Implicit deny	Y (Direct)	Implicit deny
Direct Allow	Y (Default)			
Template Deny	Y (Document Life cycle)			
Template Allow			Y (Security Policy)	Y (Security Policy)
Inherited Deny				Y (Security Folder)
Inherited Allow		Y (Security Folder)		
Marking use	Y	Y	Y	N
Marking applied constraint	N/A	N/A	N/A	Deny
<i>Effective permission</i>	<i>Allow</i>	<i>Allow</i>	<i>Deny</i>	<i>Deny</i>

Security for User A

This document's default instance security grants User A a view content permission on all new instances of our example document's class. User A is also explicitly denied this permission by a property that is assigned from a document life cycle policy. A direct permission has precedence over a template permission, meaning that the default allow permission is still valid. User A is also granted the use right on a marking that is associated with the document, so the view content permission is not denied to User A by the marking constraint mask, which results in User A having an allow permission for view content on our example document.

Security for User B

Neither the default instance security nor the document itself assigns User B a permission for the document, which results in an implicit deny, as by default, the Content Engine denies permissions unless they are explicitly granted. This document instance does have a security folder specified, explicitly granting User B the view content permission. User B also has use rights on the marking

constraint mask, which results in User B having an allow permission for view content on the document.

Security for User C

User C has an explicit direct deny for the view content permission on our document. Even though an allow view content right exists from an applied security template, this has a lower precedence than the explicit deny. User C does have the use right on the marking, but this does not grant any extra rights to the document (a lack of the use right means the constraint mask denies the right). The effective permission is therefore to deny view content because of the explicit deny User C from the direct permission on the document.

Security for User D

No direct or default allow for view content is specified for User D. Because of this, User D is implicitly denied the view content right for the same reasons that User B is. User D is, however, granted the view content right from an applied security template, so this takes precedence. The security folder has a deny view content permission entry, but this is of lower precedence than the allow view content from the security template. At this point, User D has an allow view content permission. User D does not, however, have the use right on the marking that is assigned to the document. As a result, the constraint mask is applied to User D's rights, denying User D view content rights on the document. Markings mask or remove individual permissions from a calculated permission list, so this is the setting that is applied. User D is denied view content rights on the document.

8.4 Setting security across the enterprise

When you know the default security for a particular class of object at design time, you can set it in the default instance security tab of the class. However, it is often the case that these security settings are common across document types and roles. In this situation, it is useful to store the ACEs in a reusable container that can be assigned to multiple documents. This has advantages for both manageability and reduction of the database size due to access control entries. In this section, we describe the options that are available and when you choose to use each one.

8.4.1 Marking sets

Marking sets operate by having a list of values called markings, where each has its own list of grantees who are given the *use right* by the marking. A property

can then be created on an object whose values are taken from a marking set, which works much the same way as choice lists, except choosing a particular marking value affects the security of the document.

Although the document instance security settings are still present, the marking constraints are applied or masked over these, which makes markings very useful when stringent security that cannot be overridden needs to be applied across an enterprise. A classic example of this is military or intelligence applications where there is a pre-existing security framework. A marking for a document can be created such that setting the value to Top Secret denies all users in lower security groups access to the object.

Markings: A marking differs significantly from other security methods used in IBM FileNet Content Manager. Normally, permissions that are given or denied from specified users and groups are explicitly listed. With a marking, however, only users and groups that have the *use right* on the marking need to be specified. If a user does not have this right, then their computed permissions are masked (denied) by the constraint permissions that are enabled on the marking.

There are two types of marking sets: non-hierarchical and hierarchical. Non-hierarchical sets apply only the constraints that are present on the selected marking value. Hierarchical sets, by contrast, apply all lower marking use rights prior to its own, which means that for the previous example that the *Top Secret* marking only holds additional entries for users with Top Secret level access rather than re-list all the entries for Secret, Confidential, Restricted, and Public. This makes administration much easier and prevents oversights in security from occurring.

Let us look at an example. Let us say that we want to ensure that all non-HR users cannot read the properties or content of all documents that have a department marking property of HR. We create a marking set called Department Set with several markings, one of which is called HR Department. We specify that the use right is given to the HR Department group as held in the directory server. We also add a constraint mask for all of the permissions that we want to deny. You can see, in Figure 8-3 on page 207, that we deny all permissions to groups other than HR.

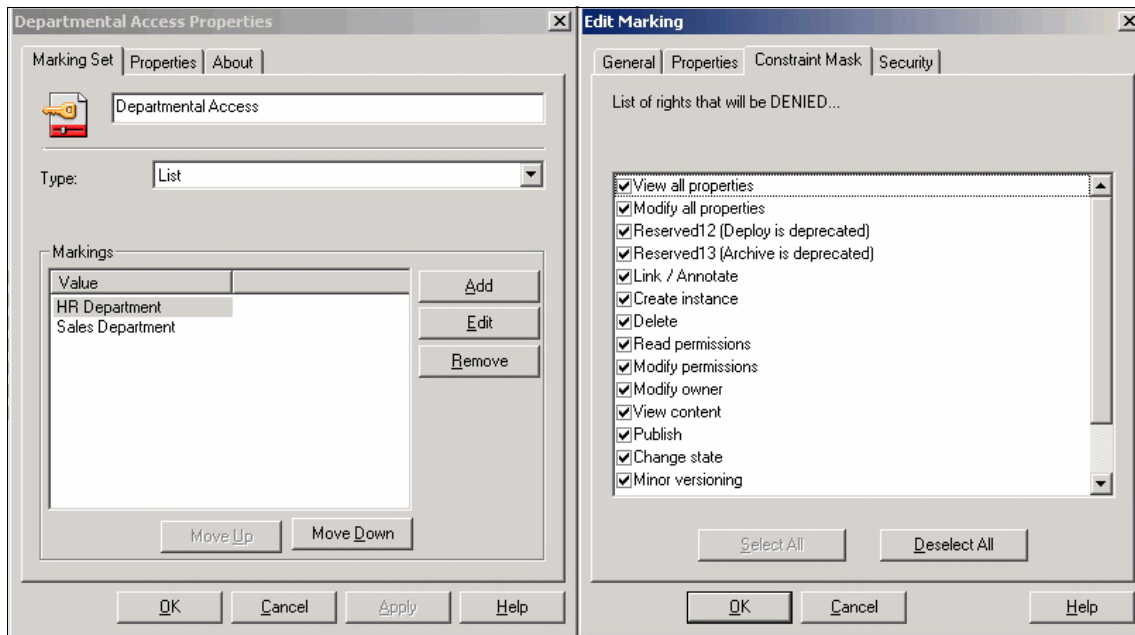


Figure 8-3 The marking set properties and marking properties dialog boxes

If a non-HR user tries to access the document, the user is prevented from reading the document's properties and content by virtue of the user not being in the HR Department group.

Implementing the equivalent security restrictions using permission lists require specific enumeration of all groups that should not view content and properties, making security management across the organization far more difficult than using markings.

Markings do not exist at the object store level. They are configured on the IBM FileNet P8 Domain and as such can be re-used by any object store. The advantage of this is that updating a marking has instant effect across the enterprise. Multiple markings can be applied to the same document, which causes all constraint masks that are specified in the active markings to be enacted cumulatively.

Now we look at the classic example of a security classification. We want to create a hierarchical marking set so that all access that is denied at the Restricted level is carried on to the Confidential, Secret, and Top Secret levels. Remember that a marking set propagates only the use, add, and remove permissions. Permissions are only masked (and thus denied) by a marking if its

constraints mask has specific permissions that are listed to deny. You can see how a hierarchical set propagates Allow and Deny rights in Figure 8-4.

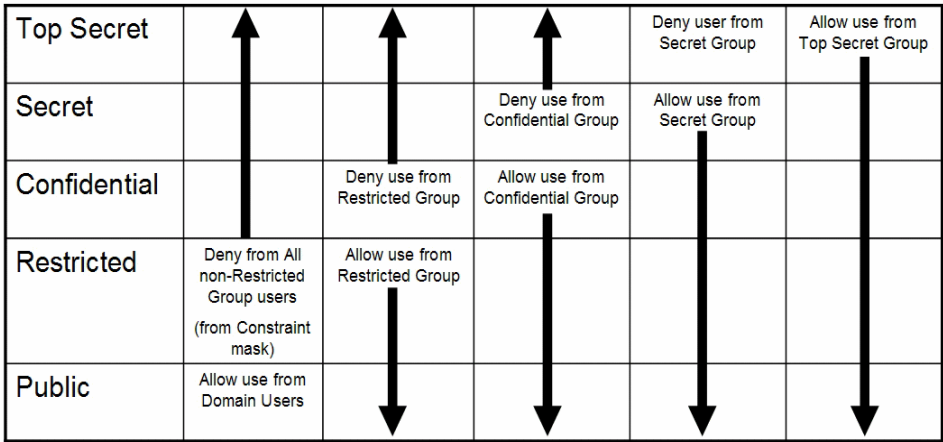


Figure 8-4 Use permission propagated in security classification marking set example

As you can see, hierarchical propagation has the effect of *deny rights being propagated upwards*, and *allow rights being propagated downwards*. In other words, if a user has use rights at the Secret level, they also receive use rights for all documents that are marked Confidential, Restricted, and Public.

On all markings other than Public, we specify all permissions in the constraint mask, which denies access to any object from anyone in the system who does not have the Use permission on the marking. For Public, we added an Allow use for all domain users. In our setup with Active Directory, all users are members of the domain users group. Consequently, we did not add any constraint mask for Public because it is not evaluated.

Markings: Markings allow content to be hidden from system administrators. A marking that denies the Modify Owner right to all users that should not change a document, folder, or their security can be specified. The administrator can be included in this group. The Use Marked Objects right can then be assigned to all other users of the system that need it. Adding a required property that uses this marking to the top level Document class (or Folder or Custom object) with the default value of the specific marking created enforces the marking for all new documents. In this way, administrator access is denied to any content within the system, while still allowing them to configure the system.

Expansion products use the core functionality of the IBM FileNet P8 Platform and apply these capabilities to new problem domains. Marking sets are used extensively in IBM FileNet Records Manager to lock down content that is

declared as a record. This marking prevents any user who is not a member of the Records Managers group from deleting or modifying the content and properties of any document version that is declared as a record. Setting marking set values can be a manual step, such as when an investigator accesses the document or an automated step during records disposition process, checking, moving (to long term storage) or deleting a record during its life cycle.

This approach provides several advantages. The security mechanism in use is familiar to existing IBM FileNet Content Manager (or IBM FileNet Business Process Manager (BPM)) administrators, which makes it easier for administrators to start using an expansion product. Many of the groups and access control specifications might already be in place or can be re-used for an IBM FileNet Records Manager implementation. A customer also gets the added assurance of knowing this security feature was extensively tested in IBM FileNet Content Manager (or BPM) implementations.

Markings: Documents can have multiple markings. This is allowed and the effective constraint mask is calculated with all cumulative denials being applied first, followed by the allows.

It is also possible that a marking property can have more than one value. A document can, for example, be applicable to multiple departments, but might not be allowed to be accessed from anywhere else. Be aware that hierarchical marking sets cannot be assigned to a multi-value property. As an element in a hierarchical set inherits settings from lower precedence markings, it does not make sense to assign multiple hierarchical markings to the same property.

8.4.2 Security policies

Security policies consist of security templates. These templates contain ACEs just like a document instance's security settings. There are two types of security policy templates: *version templates* and *application templates*. A version template is assigned to a document when its versioning state changes (for example, Released, In Process, Reservation, or Superseded). An application template, by contrast, is assigned by a custom application, explicitly setting the template through an API call.

This behavior makes security policies more decoupled from the document's properties. Permissions to apply do not need to be explicitly stated but rather change with the document's state. After this change is made, the permissions on the applicable security template are copied onto the document version. They are not dynamically resolved like they are for marking sets or dynamic security inheritance properties.

Whether the security policy adds the applicable template's entries to the document's security permissions or whether it completely replaces them can be specified. This option is particularly useful because each security template only needs to contain the changes to the document instance security that needs to be applied.

Security policies can be best thought of as objects that modify a document's security during its life cycle rather than as explicitly overriding security based on some manual action.

A default security policy can be assigned in the class definition settings dialog, as shown in Figure 8-5. They can also be assigned to a specific version of a document later in its life cycle. However, changes to the current version's security policy is only processed the next time it is versioned because that is when the security templates are checked and applied. Do not apply one to the current version and expect it to change the document's security immediately because that does not happen until some versioning state event occurs, such as Checkout.

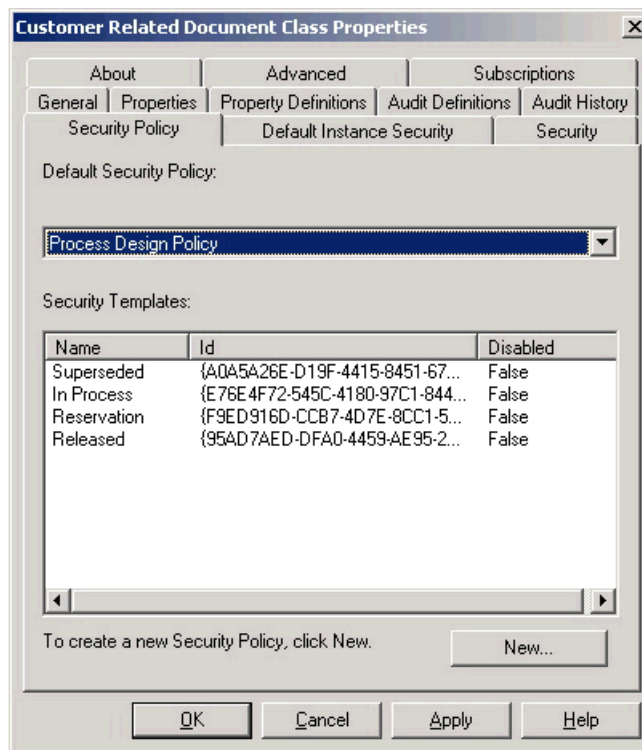


Figure 8-5 Assigning a default security policy to a new instance of a class

In addition to the four default document states, custom or application states can also be created, which are useful in situations during a document's life cycle, during a managing process, or when it is known at design time who should have access to a document.

Consider the example of a software company that produces and publishes documents. We call these documents technical papers. A technical paper can go through an authoring state when at a minor version (such as 1.1, 1.2, and 2.4), and an editorial state. After it is checked by an Editor, it can be promoted to a major version that is ready for publication. The security for these steps can be modified using versioning security templates. Consider that we want a member of the Legal team to also check the technical paper before it is published. The Legal team has access to the Released version, thanks to a versioning security template. After they check the technical paper, they move its state to the Ready for Publication application state. This state prevents any further modification to the document and requires a custom application state to be specially created and applied to the technical paper class.

Using this method, technical papers are moved through their application-specific life cycle, modifying security along the way as most appropriate. We accomplished this by configuring document states and security templates within the Content Engine and the process required no business process.

Of course there are situations where this approach falls short. An example might be when security should not be assigned to a group of people with a role, but rather it needs to be dynamically decided. This type of requirement is best accomplished within a business process, which we describe in 8.5, "Security requirement changes with time" on page 219.

There is also the issue where only one security policy can be assigned to a document at any one time. In certain situations, you might want to add extra security to a document. An example might be adding a restriction on who can see a document by country because of specific local legislation. In this case, it is best to use a marking set or a Dynamic Security Inheritance Object, depending on the situation.

8.4.3 Document life cycle policies

It is possible to create document life cycle policies to have more granular control over how a document is accessed and worked on during its existence. If you consider a simple document creation, approval, and publishing example, you can use a document life cycle to set the visibility (view content permission) of a document at specific phases in its life cycle, as shown in Figure 8-6 on page 212.

Create a Lifecycle Policy

Specify the Lifecycle States
Click Add to add each new state to the policy. List the states in the order in which the document moves from one state to another.

Sequence	State	Demotion allo...	Apply Template
1	Being Edited	N	Y
2	Under Review	Y	Y
3	Approved	N	Y
4	Published	N	N
5	Removed	N	Y

Buttons: Add..., Modify..., Remove, Move Up, Move Down, < Back, Next >, Cancel, Help

Figure 8-6 A simple document editing, approval and publishing life cycle

Security can optionally be assigned when a document transitions to a particular state. When going through approval and (re-)editing, the author's edit content rights might be temporarily removed and then added back again if a change is required. Not all states need to change security.

After a document is approved, all versioning and modification permissions are removed from both the content authors and content approvers in the previous example. No further security modifications are needed until, for example, the document needs to be checked out for an update, so the Published template does not apply any security of its own.

It might be necessary to perform some system action based on changes in states, which is done by assigning a custom Java class as a life cycle action that is invoked whenever the state changes. For our example, we might want to file the document in different locations after it is published, and then automatically move the life cycle on to the published state.

Any permissions that are assigned by a life cycle policy have a source of template, and are therefore evaluated at the same level as security policies. A life cycle policy can be assigned to one or more document classes, but each class can only have one life cycle policy.

Document life cycle policies can be useful if the processing requirement is simple and a full business process management solution is overkill.

Dynamic security inheritance

A newly exposed feature is the concept of security inheritance. Rather than have a property value determine additional security on a document, the property, itself, specifies another Content Engine object from which to inherit the security settings.

The object that inherits the security still has its own direct security settings and they are not modified using this method, but they are supplemented by any inherited permissions. If a Content Engine object has direct, default, or template permissions, however, they have a higher precedence than inherited permissions.

You specify a security inheritance object by creating an object value property and specifying its security proxy type as inherited. If the value is null, then no extra permissions are applied. You can also specify the property as being required, disallowing null values. The changes to dynamic inherited security are applied immediately and evaluated on the fly. When this property is assigned to a class, the parent class of the object from which security is being inherited must be selected, as shown in Figure 8-7 on page 214.

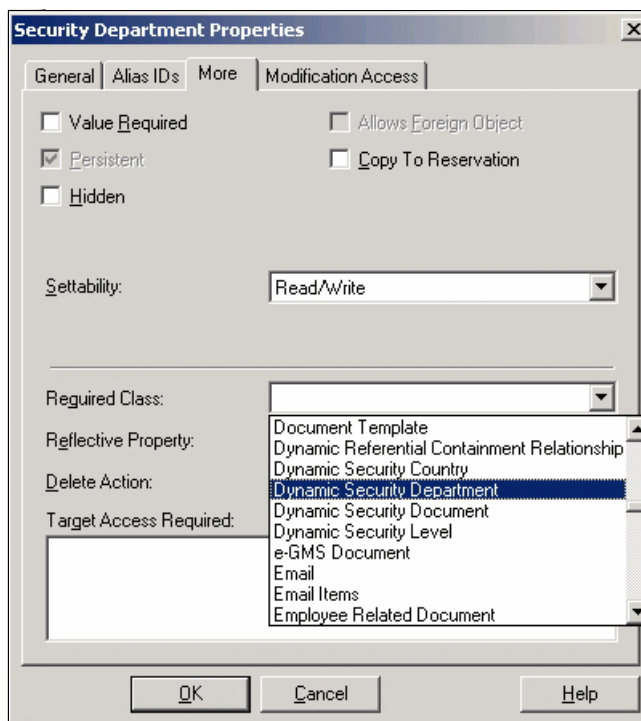


Figure 8-7 Setting the class of the object this class property is to inherit from

You can also set the reflective property, which allows security to be inherited from an independently securable object that is dependent on the target class, which can be an annotation object's security permissions, rather than the object's own permissions. You can leave this reflective property blank to just inherit the document's permissions list.

Remember that the target class is specified at the document class property definition settings level and not at the object store property settings level, which means that different classes can re-use the same property but specify that they only allow values from different delegate classes.

Note: Only permissions marked as *apply to this object and all children* and *apply to this object and immediate children* are applied to the delegating object, which depends on inheritable depth. A delegate object might also inherit permissions from another object, to any level.

You can also force users to specify a value for this property and provide a default value at the class level.

Dynamic security inheritance also works on Custom Object and Folder objects, which can be mixed and matched. You can create a folder, for example, with the target security that you want and assign this as the security delegate for a document. This means that you can now assign security permissions to one Content Engine class that are not applicable to that type, but only to other types, for example, you can add a View Content permission to a folder, which has no effect on the Folder permissions. However, if this Folder is assigned as a security delegate, then the delegating Document has this permission applied to it.

An advantage with setting the delegate class as a document or a folder is that unlike a custom object, these classes have their friendly display name shown in the Workplace and WorkplaceXT property value selection interface. Also, as you specified a particular class, going to the search page and just clicking search returns all valid objects in the current object store for that delegate class, as you can see in Figure 8-8 on page 216.

The Security Folder property: The Security Folder property is an out of the box security inheritance property that IBM FileNet Content Manager provides, which exists on all Document objects. It works in the same way as other inheritance properties, with the provision that the object that it specifies is an instance of Folder or one of its subclasses, which also means that the Document does not have to be filed in the folder that is specified in the Security Folder property.

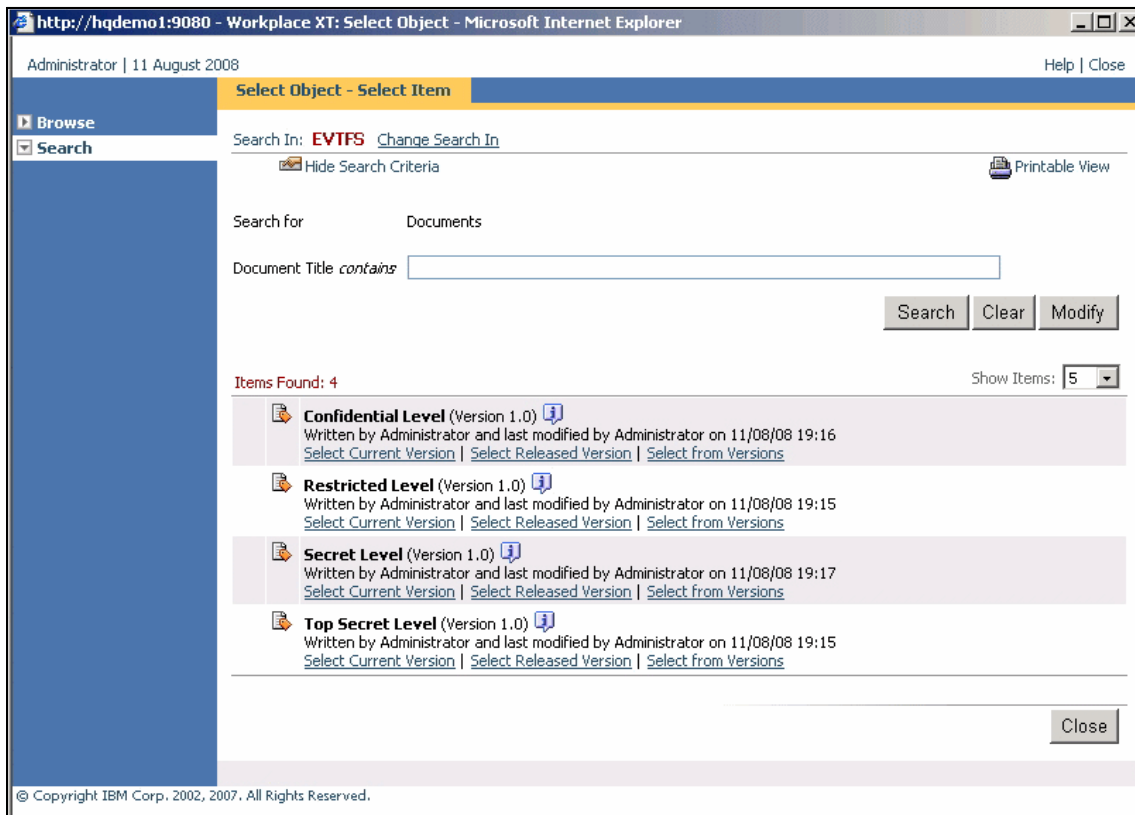


Figure 8-8 Example of searching for a delegate class

Figure 8-8 shows that the security classification is implemented as dynamic security inheritance properties instead of markings. Markings are better for this particular use case, as we explain in the example at the end of the chapter, 8.9, “A practical example: Re-insurance placement and litigation” on page 240, but this diagram just illustrates how delegate objects can be searched for easily.

These objects can also exist in different object stores, making it very easy to centrally define security delegate objects and use them throughout the enterprise. In this fashion, security can be modified on thousands of documents by modifying permissions on just one Content Engine object, enabling far easier security management than is possible when dealing with individual objects.

Another prime example of where you can use this feature is when all documents that are assigned to a particular content author must have the same security settings. It is possible to specify, for example, that the author's manager and lead editor always have access to this content, meaning that the author never has to remember to specify the security settings manually. It also means the that

security settings are created one time and used repeatedly without copying ACE entries throughout a Content Engine object store database. This feature makes handling staff changes much easier and allows users to be quickly added and removed from the delegate object and these changes to be reflected instantly across all of a user's documents.

8.4.4 Ethical wall

The purpose of creating an *ethical wall* (previously known as Chinese wall) within a company's environment is to ensure that two internal teams using the same IBM FileNet P8 system do not see each others' content, which occurs in competitive situations where an organization might be subcontracted by two or more competing firms and needs to show compliance by securing both sets of content separately. This scenario could be design information, legal disclaimers, problem reports, or contractual amounts. In such cases, it is vital that information be restricted only to the people who need to see it.

In such scenarios, it is best to restrict any privileged groups, such as *write any owner* and *privileged write* at the object store level, and then make sure that the folder that contain some group's content is locked down so as not to inherit any permissions from elsewhere in the chain, for example, you create a project level folder whose owner is the project manager. You then assign security such that only the owner can modify permissions on this object and all its children, which means that all sub-folders inherit this restriction, such that the folder owner is the only person that can set security in this work area.

You must ensure that the documents do not have any Allow permissions for other groups in the system, perhaps using Marking Sets to accomplish this. The problem with this approach is that over time, the marking set increases in size. As a result, every project, including perhaps those that were complete for years, have a marking option set in it. It also means that one project team member has access to a list of all projects within the system! Unless the marking sets are updated whenever a new project is added, the administrative overhead can become prohibitive because it requires modification of the IBM FileNet P8 domain, which is where the markings exist.

Implementing an implicit Deny through not assigning any explicit Allow permissions to people not in the project team is a risky business. It might be best to implement a marking for a project team to be used while projects are active, and have another marking property for locking down the content after these people stop working on the documents, which can be done by declaring these documents as records, which makes IBM FileNet Records Manager apply its own restrictive markings to the content.

This approach can offer the best of both worlds, providing a short to medium-term project team marking that can be updated as project teams are formed and dissolved. Declaring these documents as records prior to the project team being dissolved would further prevent unauthorized access to the content.

Note: In general, when access to content needs to be denied to everyone except a particular set of groups/users, then consider using markings.

8.4.5 Creating a shared service

Many organizations purchase emerging technologies at the departmental level until a critical mass is reached and it makes sense to deploy the technology centrally. A classic example of this is database applications. Many organizations started deploying databases as standalone storage mechanisms for application information and configuration. Databases gradually became more ubiquitous and managing the information, security, backups, and resilience to failure became a major challenge.

Most organizations now have central teams to look after mission critical software, such as a database farm. As Enterprise Content Management systems become more common, we are seeing the same trend of adoption. It might be that forward thinking departments are setting up a team to look after a service even while it is deployed at their department's level, which enables them, in the future, to allow other users in the wider organization to have access to the same information, creating a single source for information stores. A classic example of this is customer information that is managed in different departments, such as e-mail support correspondence, account opening information, and billing statements.

It might also be the case that an organization wants to create a shared service while also maintaining independent, discrete sets of information. A fraud investigation team, for example, might need access to the entire organization's set of customer information, while separately maintaining its own secured repository for investigation reports. Such a scenario requires a solution at the object store level.

By using the same IBM FileNet P8 domain for different applications, it is possible to gain the advantages of not requiring any additional server or software rollouts. Creating an extra object store requires the creation of a new database table space and configuration of some data sources, but after this is done, offers the full benefits of a secured repository that uses corporate-wide security settings from marking sets while implementing new application specific settings. It might also be that you do not want the wider organization to see certain types of information, represented by Content Engine properties, that is stored with your

documents. Creating a separate object store can hide from the wider organization the fact that these properties exist.

A separate object store is also very useful when it comes to calculating how much resources departments are using at both the database and file storage levels. Because the entire application has its own storage locations, calculating the space used, and therefore the amount to charge that department, becomes very easy. It is also much more transparent to the department that uses the shared service.

There is an easier way to deny access to all objects in an object store without using marking sets. You can deny the *connect to store* permission on the object store to anyone who should not have access to any object in the store. If you wanted, instead, to allow only read access, you can deny the *modify existing objects* or *delete objects* permissions. This option has the added benefit of not requiring any change to the metadata model, as required with marking sets. Also, because marking sets are used across object stores, they cannot be used to deny permissions to just one object store; instead, you create object store-specific marking sets and assign them to classes, which diminishes the advantages of using marking sets.

8.5 Security requirement changes with time

Organizations are very fluid environments. Staff changes are common. Documents might need to be restricted or made available to different audiences. Over time, business processes are modified, types of work are added, and content lifecycles might change. There is a need to respond to the changing security landscape while ensuring that any existing information remains protected.

In this section, we describe the various issues that are related to a changing security environment and discuss how the IBM FileNet P8 Platform can be used to minimize the administration effort.

8.5.1 How document and process life cycle affects security

As we learned in the discussion in 8.4.2, “Security policies” on page 209, changes in documents during their life cycle can have implications for those who can access a document. There is also the question of how to manage the assignment and removal of dynamic permissions. Take the example of a customer sending in a request to open a bank account. This information is not made publicly available to anyone in the organization. It might be that at certain points in the management process, an account manager is assigned. After the

account manager is assigned, that individual needs assigning permissions to the content for their customer.

In the examples provided so far in this chapter, we can cover this use case by assigning an application security policy template for a state of Account Manager assigned. However, your business requirement might want only individuals who are acting in a particular role to have Add access to the content.

Business process management is very effective at retrieving and acting on information in real time. For a particular type of account, or the workload given to particular users, we can automatically decide which user to assign as an account manager. You can also assign a supervisory user within the Account Opening process to select the appropriate person to be the account manager. In either case, the information that concerns the person who should be assigned is present as part of the process, and it is the process that needs to act upon it. No document event happened to necessitate this change. It is purely a function of a management process reaching a particular stage.

There are many ways to secure this information within a business process management environment such as that provided by the IBM FileNet P8 Platform (specifically, IBM FileNet Business Process Manager). You use the user interface to restrict who can see a work item. You use the concept of process attachments within a process and security on the queues to restrict who can see information. Although these are valid methods, to be totally sure that content access is secured, we must ensure that the underlying security permissions are set, which is the only way to absolutely ensure that a user without permission cannot see the content. This feat is possible within IBM FileNet P8 because all expansion products are built upon the Content and Process engines and are restricted from what they can see and process by virtue of those engines enforcing their security models onto the client application.

This ability brings up a range of questions. How can we lock down queues and individual work items? How can we dynamically interact with the Content Engine in order to update security in real time? How do we manage the updates or changes to such activities? How can we audit such activities? What are the drawbacks? We answer these questions in the remainder of this section.

8.5.2 Managing security updates

To manage the security of an individual document, or class of documents, it is relatively simple thanks to the rich features that are available in IBM FileNet Content Manager. After you start thinking at a more abstract level, things might become more difficult to manage, for instance, let us say that you have 300 client applications being worked on in a given week. If you need to retain these

applications for five years and you get 100 new ones per week, you have an additional historical set of 26,000 to manage.

Now let us say that you have a security policy that manages documents while they are in use. The security settings are assigned to the documents while they are being modified, or versioned, or at least by the application. So if we now change our security policies within the organization to reflect a change in how we do business, we need only modify the single security policy. Because we have 300 documents in process, all of these documents and new documents in the system have the policy applied the next time they are versioned. If the 26,000 historical records are relatively static, we have an administrative challenge to handle because the security on these will not be updated to reflect the new policy.

Conversely, if we instead use marking sets to manage this security, changing this is immediately reflected by all documents within the system. However, the historical documents' security no longer reflects the security that was originally assigned to them at the time of processing, which can be a problem when trying to prove to a regulator that the security on a document, during or at the end of a process that, was used to manage it.

Another problem with updating marking sets is that if you remove or add a marking, it is not reflected on any documents that currently use the marking set. Removing a marking from a document because it was removed from a set does not make sense because it could leave the content open. As such, the marking still applies to a document in the Content Engine until that the document is versioned or the property is modified, which still leaves us with a management overhead if we add and remove lots of markings, rather than just modifying the underlying permissions that they assign to content.

Table 8-5 describes the various types of security, the ease of modifying their permissions and adding/removing elements, and therefore the longevity for which you should consider using them in a production, long-lived system.

Table 8-5 Advantages and disadvantages of various security methods

Security Method	Ease of updating permissions	Ease of adding or removing elements	Relative longevity
Direct assignment	Easy for individual documents, but difficult for large groups of documents	Easy	Short term. Might require changing many times during life cycle. Useful for short term, dynamic assignments of permissions.

Security Method	Ease of updating permissions	Ease of adding or removing elements	Relative longevity
Dynamic Security Inheritance Objects	Easy for large or small groups of documents, if using a Choice List to limit selection. Instantly applied to all content. Requires quite detailed system knowledge. Can be used across a IBM FileNet P8 Domain.	Easy. You cannot delete an element if it is in use, but you can hide it from being selected to prevent its use in the future.	Short to long term. Can be used effectively by business processes for application-specific security settings.
Marking Sets	Easy. Instantly applied to all content. Can be used across a IBM FileNet P8 Domain.	Difficult. Leaves you with issues around older documents.	Long term. Use for rarely changing, enterprise-wide security types where permissions might change but the number of types of access do not change often. Use multiple marking sets if longevity varies considerably.
Security policies / Document life cycle policies	Medium. Easy to change, but is not applied until content is versioned or actioned in a custom application, for example, its state changes. Can be assigned at instant creation time.	Easy for version states, but difficult for a custom application because it requires coding in the application.	Medium. Useful while the document is in use because it abstracts the individual permissions from the document. Not good to lock down content over long periods of time.
Security folder	Easy and instance, thanks to inheritance.	Medium. If a document is filed in multiple locations, this can confuse users as to why security for one folder is not being applied. Also there is no simple interface in WorkplaceXT to assign a security folder.	Short to medium. Useful while document and folder are in use, but difficult to maintain over time for retention purposes.
Default instance security	Medium because it is an administrative task. Only applies to new documents.	Not applicable. This applies to one per class.	Short. Only used at document creation. Useful to initialize owner, policies, dynamic security inheritance objects, and marking set values.

As you can see from Table 8-5 on page 221, all methods of assigning security to content can be very useful but only applicable to specific use cases. To build a comprehensive solution to a problem that requires content to be managed over time, a mixture of methods often needs to be used.

In the remainder of this chapter, we discuss how to perform changes over time. The example at the end of the chapter pulls these short to long-term methods together in a real world example to illustrate how a platform approach to these security issues can assist the management of information security.

8.5.3 Update using business processes

It is often a requirement to update a document's security over the course of its life cycle. The examples that we provided so far are either manual or controlled by document life cycle or security policies. There are other scenarios, however, where security might need to be updated dynamically in response to user decisions or information from an external system.

The Process Engine (offered through IBM FileNet Business Process Manager) can be used to execute component steps that call Content Engine API functionality to update security on Content Engine objects. An example of this is when a new customer requests a product and they must be assigned a dedicated Account Manager. We might not want anyone else to see that content, and as such the business process has to allow the Account Manager that the Sales Administrator selects to see the document, which means that the Sales Administrators do not need to remember which permissions in the Content Engine to assign because they simply select the appropriate manager, and the business process handles the rest.

The out of the box CE_Operations component does not have any security querying or modification functions, but they are simple to create. Consider Example 8-1.

Example 8-1 Sample code to add grantee level

```
public void addGranteeLevel(VWAttachment attachment,String
username,String level) throws Exception {
    Document doc = findDocument(attachment); // utility function

    // create a new access permission object
    AccessPermission ap = Factory.AccessPermission.createInstance();
    Integer l = getAccessPermission(level); // utility function

    // set access permissions
    ap.set_GranteeName(username);
```

```

ap.set_InheritableDepth(new Integer(0));
ap.set_AccessType(AccessType.ALLOW);
ap.set_AccessMask(1);

AccessPermissionList ap1 = doc.get_Permissions();

// add the permissions to the list
ap1.add(ap);

doc.save(RefreshMode.NO_REFRESH);
}

```

Example 8-1 on page 223 uses the Content Manager 4.0 Java API to find a document, create a new Access Control Entry, and save the changes. The access mask is computed by adding together the integer value of all permissions that we want to give to the grantee (user or group). A utility function is used in Example 8-1 on page 223, so process developers can simply supply *View Content* to specify the level, rather than remember the correct integer value for each of the constituent permissions.

Now that we secured access to the documents throughout a process, we must ensure that each step is accessed by the appropriate person. We already mentioned that Queues and Rosters can be restricted to particular users, but we also have the ability, with IBM FileNet P8, to assign work to specific users and workflow groups. A workflow group is an array containing references to users in the system. A workflow group is completely different from an LDAP group and is only relevant to a particular process instance. Assigning a step to a workflow group means that each of these users are sent the step to work on. After these users all complete the work step, the routing condition is interpreted as usual.

It is possible to dynamically assign strings that contain user names to elements within a workflow group, which means that for our example we can map the assigned account manager string field to a workflow group and then route the process to a step that is linked to this workflow group. In other words, even though at process design time we had no idea who would be assigned to do this work, we can still effectively design and transfer our process while guaranteeing this information is provided for each process instance by a previously assigned user, which makes organizational changes easy to separate from business process design, thus maximizing the flexibility of the IBM FileNet P8 Platform.

8.5.4 Critical records declaration, holds, and disposition

Within the Content Engine, itself, there is no built-in concept of time-based events, only document events that occur by virtue of an action being performed

on a document. The most common use case for actions that occur on content after a certain period of time is records management. A document can be declared as part of one or more records within the system. These record types have their own policies with regard to review and disposition cycles. This information that is stored within the Records Manager object store is periodically checked and validated against the records within the system by the Records Manager Sweep.

The Records Manager Disposition Sweep is an application (provided by IBM FileNet Records Manager) that checks to see which records are ready for review or disposition. For each record it finds that meets the specified period of time, Records Manager initiates the appropriate management business process. These processes are naturally extended to fit the type of record and organizational procedures that exist for record review and disposition. After a review process is complete, it might be decided that a record is no longer required and should be deleted.

The sweep finds these documents that are marked for disposition and verifies that they should be deleted. A document can be declared as part of multiple records, so deleting it while it is still needed for another record process is not allowed.

It is this concept of record types, each of which has its own retention schedules and actions, that makes IBM FileNet Records Manager such a powerful solution for automating the management of critical business records. Content can be automatically, intelligently, and consistently assigned to one or more record types through records declaration. The support for a file plan, that is, a conceptual hierarchical view of content as discrete record types, gives a unique and useful view of content that exists within the repository.

The ability to locate records of multiple documents, to place extra security on them using declaration and legal holds, to find them easily, and manage their full life cycle through sweeps, is why IBM FileNet Records Manager, the expansion product, is so useful. It uses the inherent features of the IBM FileNet P8 Platform to provide a rich information life cycle management paradigm that makes managing and enforcing business and regulatory rules extremely simple using a capability known as Zero Click records declaration.

For more information about IBM FileNet Records Manager and the security-related issues, refer to *Understanding IBM FileNet Records Manager*, SG24-7623.

8.5.5 Institutional reorganizations

People who move internally or join and leave an organization can prove to be a real security management challenge. This challenge is especially true of systems that assign security to information that is based on an individual rather than just a role. Take the example of an Account Manager who might have document instance security set such that only they can see their customers' documents. If this person leaves, how do we update that security?

The answer lies in the mix of security that is in use. If cross-document and cross-enterprise methods are used generally for all content that is not in process, then the only documents we must worry about are those that are active. In this case, the documents in question are likely to be one of three types:

- ▶ Any personal documents for that employee, such as HR information, or a personal 'My Documents' type store.
- ▶ Business role-specific documents. In our previous example, this is documents that are related to customers and deals that the person leaving was working on.
- ▶ Documents to which the person has temporary access, such as content to review.

The first two groups of the documents are represented by specific classes of documents within the Content Engine; therefore, you can quickly build a process to find all folders and documents where the person leaving was the owner. You can do this by executing a search within a business process and then presenting the list of documents for review. After the review is complete, a new person is specified and security is updated dynamically.

You can be even more sophisticated in your approach by creating a query that returned all documents where a specific user was mentioned in the access control for the document instance. This process involves many joins across tables and performs poorly. This might be an option when an automated process is performing this outside of peak hours, but a better approach might be to avoid this situation where possible.

So, if we intelligently assign the owners of documents, we can quickly find and update the security. For HR documents for that employee, you might choose to do nothing. After all, if you declare those documents as records, they have a retention cycle to manage their disposition. All you might need to do is create a step in your "Content Re-assignment" process to update the record information and move it to the next stage in its life cycle, which then leaves just the third type of document access: short term access to documents that a person does not manage, but needs to view to perform a specific task. Typically, these are content review and approval types of tasks. In this case, there is a live process

running to manage these activities. If the process assigned this task (or step) only to this user, we have a bigger problem—how do we reassign that step dynamically?

The best approach in this situation is to do one of two things when designing your processes. The first option is to assign a step to a particular user, then always wrap that step in an escalation timer that escalates the task after a period of time. This escalation submap is reusable throughout the organization and can be constructed to lookup the current assignee's manager and re-assign the work to them. Alternatively, it can route the work to a queue instead of an individual.

The second option involves never explicitly assigning a work item to a person but rather restricting who can see and operate on that work item. You can use an application, such as Business Process Framework or the Inbasket concept in the Process Engine in IBM FileNet P8 4.5 to restrict who can see a work item based on a filter. You can then use two filters, one so that the individual who needs to work on the item sees it in their Inbasket and a second one so that the manager has an overview of all items assigned to their team. This way they can manually reassign work that they know will not be fulfilled. This process is also useful to manage loads on employees or for re-assigning work while they are on vacation.

In summary, it is best to assign security using long-term, centralized objects, such as Marking Sets, Security Folder inheritance, and Dynamic Security Inheritance objects. When content is active, its owner should be set to the most logical choice of user. A reassignment process might be necessary to automate reassigning the owner field on the document to an alternate worker. Processes also need to be designed with change in mind, with at least a compensating timer escalation to make sure work does not disappear and require manual administrator intervention to reassign.

8.6 Content-level security

After the system verifies that a user is authorized to access a piece of content, be it properties on a document or the content itself or a workflow item, we must also ensure security wherever that information flows. In this section, we briefly describe methods that are currently in use that can be leveraged. It is strongly advised that anyone who is interested in this section also seek the advice of a FileNet Certified Professional or Solutions Architect to discuss specific security requirements. There is a plethora of internal security features, such as encryption and storage of security keys and credentials, that we do not mention in this section, but that might interest you.

8.6.1 Local copies on users' machines, client cache files

A major problem for organizations is preventing duplicate, local copies of content on individuals' machines. This problem is especially important when introducing a new enterprise content management system because there is always some resistance to changing working practices by storing documents in a central system rather than keeping personal copies. The quicker this is resolved, the easier ensuring compliance becomes.

Various user access methods are possible with the IBM FileNet P8 Platform, which include thin applications, such as Workplace and Business Process Framework, their viewers such as the Image Viewer, or integrated desktop applications, such as Microsoft Office.

Whenever a browser accesses a Web page or download, the application makes a temporary local copy. It is advisable, therefore, that if the security of information is paramount, you consider using controlled access methods rather than allowing general download access for content.

The Application Integration capabilities of Workplace and WorkplaceXT can be leveraged to bring content under control while providing a rich and intuitive user experience to encourage adoption of the new system. Application Integration embeds a set of IBM FileNet P8 menu options in Microsoft Office applications to open, check out, and check in documents from an IBM FileNet P8 repository.

In recent versions of IBM FileNet P8, this feature comes installed with the File Tracker application, which monitors all documents that are downloaded using Workplace or WorkplaceXT and Office Integration. It provides extra usability by tracking the repository object that a local file represents. This tracking makes checking in modifications to content very fast because the system does not have to ask the user which document to check in.

The File Tracker has some extra, centrally configured settings that allow administrators to specify when local copies of content are deleted, which are configured in the Workplace or WorkplaceXT Site Preferences page, as shown in Figure 8-9 on page 229. The most common scenario is to remove a local copy of a document when it is checked in to the content repository. This action ensures that there is always only one current version of a document within the organization.

File Tracking	Default Settings
Enable File Tracking	No
Document directory	My Documents
Qualified path	
Delete local copy	No, Editable
Open File	No
Save As	
Select access roles	Application Engine Administrators remove

Figure 8-9 File tracking options to ensure local copies are deleted

In IBM FileNet P8 4.5, an additional Microsoft Office 2007 integration is introduced that works in much the same way that Microsoft Sharepoint integration works for Microsoft Office applications. In this scenario, all content is accessed and worked on directly from the underlying repository.

8.7 Network security

There are many issues to deal with when accessing an application over a network, especially if that network is potentially insecure. In this section, we detail the issues and how they can be mitigated, thanks to the features of the IBM FileNet P8 Platform.

8.7.1 Demilitarized Zones (DMZ)

The classic approach to network security is to have a secured server layer that only allows access from specific machines in a Demilitarized Zone (DMZ). Clients can only talk to the machines in the DMZ, and the machines in the DMZ can only talk to specific machines in the server using specific protocols and ports. Such segregation means that the clients cannot directly access the protected servers behind the DMZ.

The IBM FileNet P8 Platform supports this methodology and can be configured in several ways. Let us first take the example of a simple departmental system without any high-availability requirements. Access is made directly by a client using a browser or Microsoft Office client, which allows them to flow through the WorkplaceXT Web application. This application can be installed in the DMZ with the Content Engine and Process Engine servers located in the server layer.

On the opposite end of the scale, you might have a highly-available environment with clustering that is protected by Single Sign-On and secure Web proxy products.

The advantage here is that only the secure proxy is exposed in the DMZ. This method is used across many applications and not just IBM FileNet P8 in typical implementations. As such, there are fewer opportunities for compromising security in the DMZ layer because all application servers are located in the server layer. Having a highly-available environment also means that clients connect through one IP address that points to a (typically hardware) load balancer, which makes configuration of firewall rules easier to manage.

In the server layer, itself, the Application, Content, and Process Engines communicate to each other over known ports and protocol.

8.7.2 Encryption on the wire

It is often desirable to ensure that all clients are accessing content through a secured channel. Inside the DMZ and in the server layer, we can be confident that we are protected. On a network with a large number of potentially compromised computers, however, extra steps are needed to prevent unauthorized access to content.

All IBM FileNet P8 Platform Web user interfaces are supported in application servers that use the Secure Sockets Layer (SSL) technology, which is desirable in non-Single Sign-On (SSO) environments where users enter their user name and password directly into a Web page. Many high-profile security breaches occur in situations where the login page of a site was not on a page that was protected by SSL and the target service was. It is a good idea to ensure that SSL is always used.

SSL can be configured to provide three important data security features:

- ▶ The initiating handshake is used to identify the client that is sending the request.
- ▶ SSL can be configured to validate that the data was not modified in transit.
- ▶ SSL provides data confidentiality by encrypting data over the wire.

Behind the scenes in the server layer, you might also want all elements of the IBM FileNet P8 Platform to communicate securely with each other. This communication includes authorization lookups to the underlying Directory Server through LDAP and interaction between the core engines, such as WorkplaceXT and the Content Engine. SSL can be used to encrypt these internal communications.

It is important to note that some communications that the IBM FileNet P8 Platform uses rely on the security support of the underlying systems. The JDBC database driver, for example, needs to support secure communications with the chosen database vendor's product; likewise, the application server's JAAS login module requires configuring to perform authentication requests using secure channels. While this is out of scope of the IBM FileNet P8 Platform, it is worth mentioning that to ensure as secure a system as possible, security all the way down the application stack into the database and storage layers is also an important, and often overlooked, consideration.

8.7.3 Web services security

All Web services APIs for both the Content Engine and Process Engine are supported over an SSL protected connection. By default, all information that is known to be sensitive, such as user names and passwords that are sent to the Content Engine, are encrypted using asymmetric public private key pairs. These keys are generated during installation of the Content Engine. See the IBM FileNet P8 Platform Installation Guide for more information.

The Content Engine EJB transport options, which include RMI-IIOP and T3 (Weblogic), both support the use of SSL to protect communication. Communication through the Web services APIs can also be secured using SSL protected HTTP (https).

The Process Engine Java API connects to the Process Engine directly using RMI-IIOP, and for this API, the channel does not support SSL-protected communications. Network-based encryption techniques that preserve the IP packet must be used if this communication needs to be protected. Because all login and document retrieval requests are processed through the Content Engine Java API, this is only an issue if some of the process fields contain sensitive information because session tokens are transmitted encrypted.

IBM FileNet P8 processes can invoke, receive, and reply to Web services calls. The security aspects of these processes are important to discuss. The standards supported by the Process Engine's support for Web services are:

- ▶ WS-BPEL for managing the orchestration conversations with services
- ▶ WS-Security for passing authentication and authorization information to those services.

IBM FileNet P8 can interact with any Web services that are protected using SSL communications. This communication is handled by the WSRequest component queue that is installed in the Application Engine. The incoming messages are handled by a servlet that is configured in the Workplace or WorkplaceXT applications, depending on preference. To protect all incoming Web services

messages requires configuring at least one Workplace instance to be SSL protected.

Providing credentials within an IBM FileNet P8 process requires some manipulation of the Web services partner link variable. Doing so requires changing the underlying XML header to include WS-Security information to the target Web service. This is a very flexible method and is the same approach that is used to dynamically change the host target for your Web services call, which can be useful in situations where you want to offload the Web services call to a different geography or service provider.

In Figure 8-10, we pass in two process fields, MyUserName and MyPassword, to generate the Web services security header.

```
"<wsa:EndPointReference xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
  <wsa:ReferenceProperties>
    <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">
      <wsse:UsernameToken>
        <wsse:Username>" + MyUserName + "</wsse:Username>
        <wsse:Password>" + MyPassword + "</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </wsa:ReferenceProperties>
</wsa:EndPointReference>"
```

Figure 8-10 Providing WS-Security credentials for a Web service

For incoming Web services requests, the process can be instructed to validate the incoming user name and password information against a specified set of user accounts. The Process Engine then checks with the authentication provider that IBM FileNet P8 leverages to ensure that the user name and password are valid, as shown in Figure 8-11 on page 233.

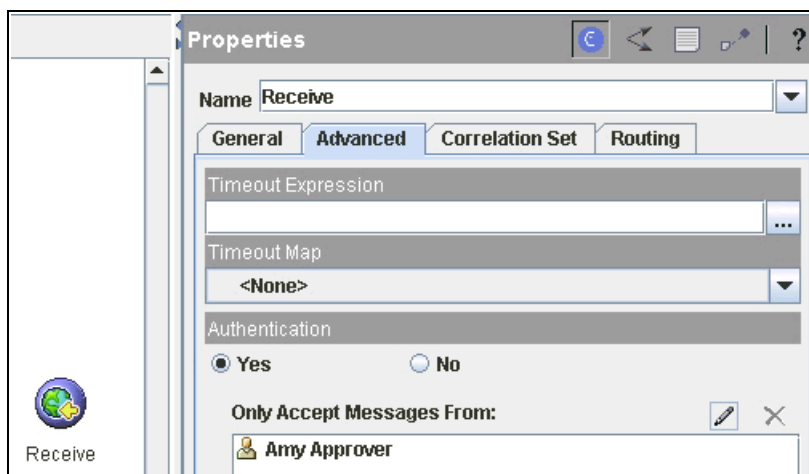


Figure 8-11 Configuring incoming authentication for a Web services call

Security can only be provided to invoke external Web services that store the authentication information in a WS-Security header. Some technologies, such as protected .NET 2.0 Web services, do not use this method and instead protect access by performing HTTP header manipulations and handshakes. These methods are not supported by the IBM FileNet P8 Platform. To interact with these services, you must enable WS-Security header support rather than restrict access to the service through HTTP authenticate handshakes. It is also relatively trivial to create an unprotected .NET 2.0 Web service to accept the incoming request, extract the credential information, and invoke the target-protected Web service using HTTP headers. This works similarly to a proxy for Web services.

8.8 Reporting and auditing

In this section, we briefly describe the kinds of logging and auditing that the IBM FileNet P8 Platform supports, its core engines, and the most common expansion products. We aim to show the depth and breadth of information that is available both to security professionals and business managers to inform their activities and prove when and why particular actions are taken in the organization. These events range from very small, atomic events, such as changing a document property, to major events that occur over the life cycle of a document or the course of a business process.

8.8.1 Logging into the Content Engine

As mentioned in Chapter 7, “Enterprise content management” on page 141, the Content Engine supports auditing of most actions that happen to Content Engine objects, which includes all actions that can occur to a document, custom object, or folder. These events can be subscribed to facilitate *active content* processing, and can also be configured on a per-class basis to write audit logs for specific, configured actions on important content. You might choose, for example, to log all document check in and delete events in the repository. For certain critical document types, you might want to enable logging of the viewing of properties or content of a document to be absolutely certain about who had access to content and when.

It is possible to create your own event types and enable logging for those too. The log information is represented as a subclass of the Event object within the Content Engine, which means that it is stored as a row in the underlying object store database. As with any other object, you can search for instances of these audit items and retrieve their properties, which you do through either the standard Content Engine search support or using the read-only JDBC provider.

Log files can also be created to log these events, lower-level debugging, and diagnostic information for the Content Engine itself. These are typically persisted to the underlying operating system logging subsystem. In Microsoft Windows, this is the application log that is accessible through the event viewer. On UNIX systems, it is stored as per your systems' syslog configuration.

The precise settings for logging can be configured using the IBM FileNet Enterprise Manager tool, as shown in Figure 8-12 on page 235.

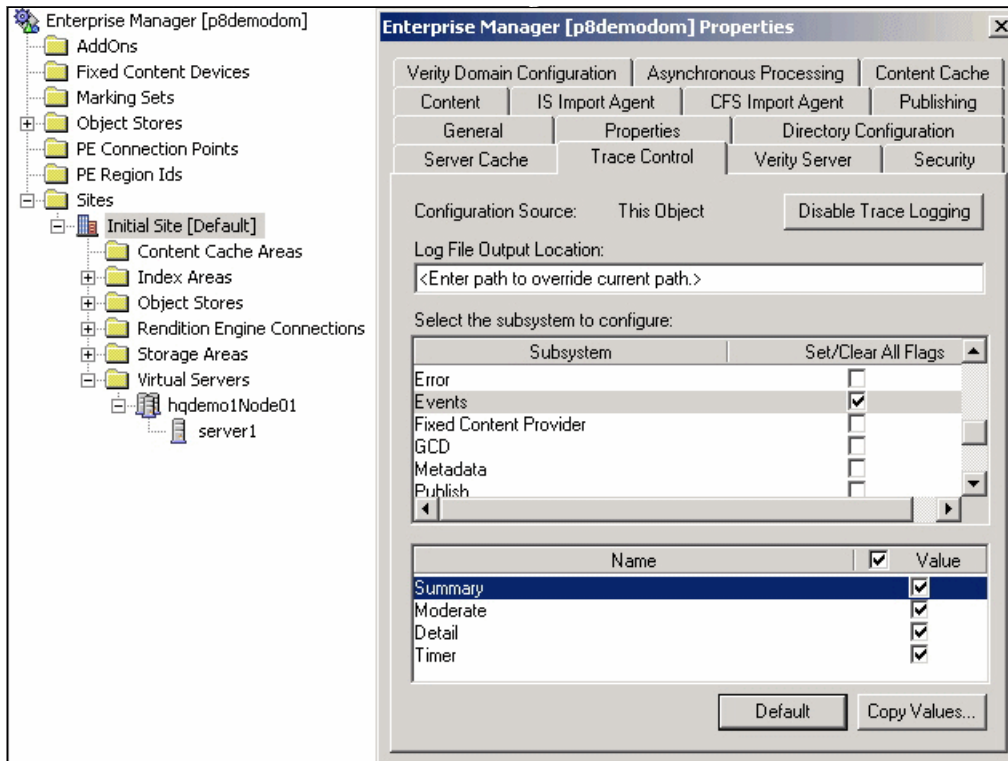


Figure 8-12 Content Engine logging configuration

As shown in Figure 8-12, logging can be configured for a variety of subsystems at the Domain-wide level, which can be overridden at lower levels: site, virtual server, and server instance. This type of logging is particularly useful when legislation requires you to implement more logging in particular jurisdictions (sites) or when you are trying to locate a performance or system-integrity issue on a specific server.

8.8.2 Logging into the Process Engine

The Process Engine log is concerned with all life cycle actions that happen during the course of a business process, which includes workflow creation, step completion, and exception handling. These events can be logged into a particular log database table. Multiple logs, each with its own database table, can be configured for the same Process Engine region, which allows the use of one log, for example, for several processes that make up a single application.

Each log that is shown in Figure 8-13 can be configured to record additional user-defined process fields to produce logs with more contextual information. Process Analyzer uses this information to add user-defined fields to the data mart, which we discuss later in 8.8.3, “Process Analyzer” on page 237.

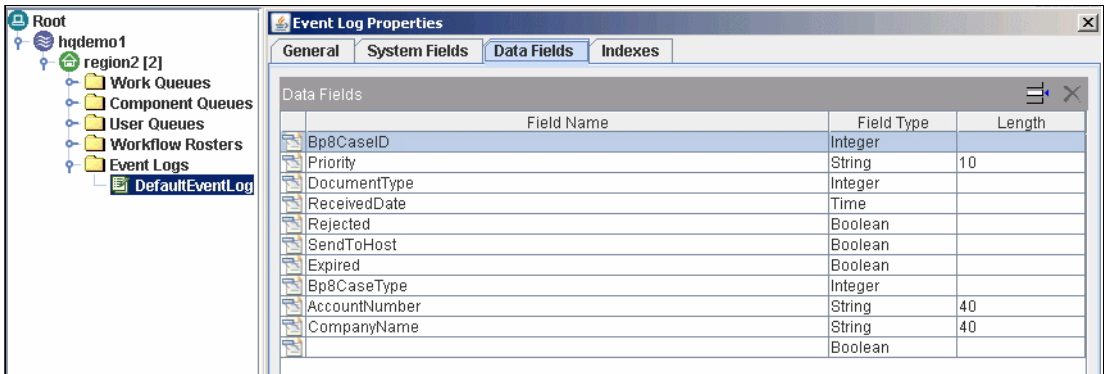


Figure 8-13 The default Process Engine log with extra configured user-defined fields

The type of Process Engine events to be logged are configured at the region level. These events can be out-of-the-box events or custom, user-defined events. These user-defined logs can be populated using the Log step type within a business process map. You pass in the event log to use, the custom log message identifier, an integer, and the message to log. The Process Engine then handles the collection of the additional, configured process fields and logs the entire message in the specified log file.

Figure 8-14 on page 237 shows the events for which logging can be enabled and disabled, where you can see that the Process Configuration Console application is used to configure both the event logs and the messages that should be logged.

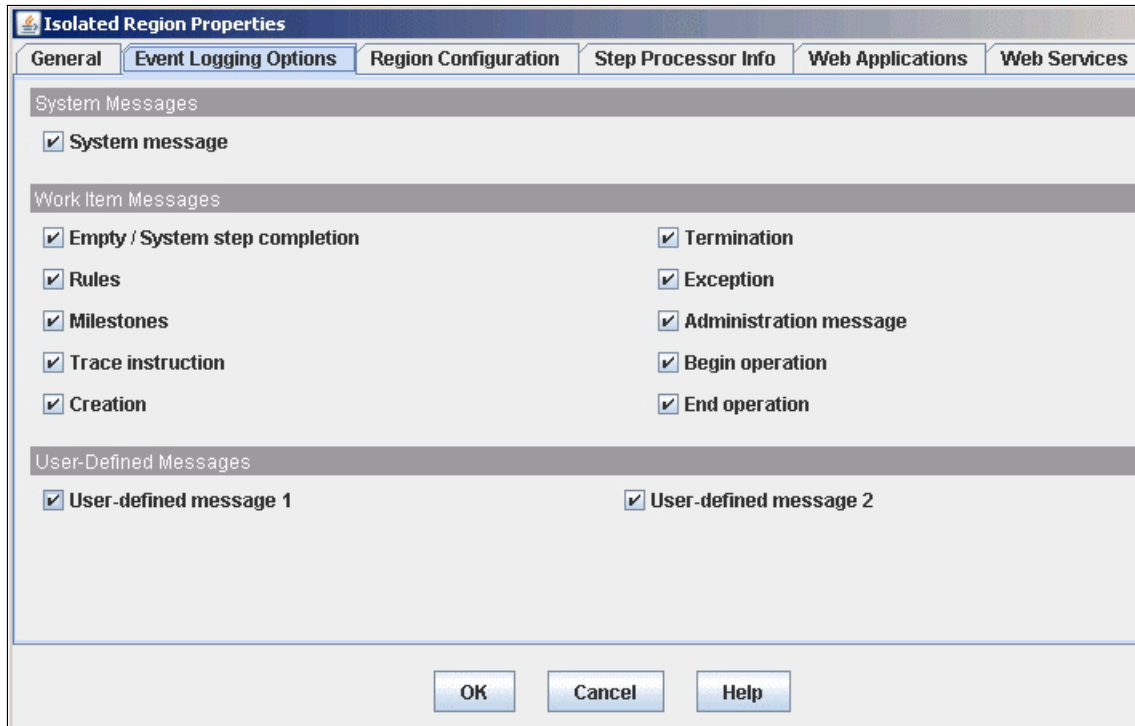


Figure 8-14 Event log configuration in the Process Engine

This information is secured by preventing any user without access to the corresponding process roster from accessing any of the log items that correspond to it.

8.8.3 Process Analyzer

The Process Analyzer extracts information from the Process Engine logs to build a data mart that contains information about what happened during processes that are executed in the organization. This information is typically about who completed work, how long it took, and what decisions were made. This feature allows Business Intelligence reporting tools to query the data mart and produce reports at either a high level or with a more narrow focus. A report can be produced to summarize the total numbers of sales by region, for example, but you can also produce a report to see how long each employee took to complete each specific type of step that they worked on.

In addition to this standard summary data, it is possible to configure user-defined fields to add additional matrix and dimensions to the reporting output. By default, these process-specific fields are not imported into the data mart.

To illustrate, consider a situation where an organization is being accused of breaking a service level agreement with a customer. In such a scenario, reporting over all processes that link to this customer is necessary. A status field determines that in fact the SLA time was exceeded, perhaps not because of poor customer service, but rather because of delays that result from waiting for the customer to provide additional.

8.8.4 Content Engine object-level JDBC provider

When the Content Engine was redesigned in Java in the 4.0 release, it was realized that it was still necessary to provide a way to get a direct object-level, read-only connection to the underlying database. By object-level, we mean the ability to query for specific document classes with *where* clauses that specify property values as opposed to database schema-level queries. Database-level queries are complex and are not logical to people who are familiar with the object-oriented model that the Content Engine provides. All document metadata for every type of document and every version, for example, is held in a DocVersion table. If all that is needed is the latest version of a specific class of document, using the object-level query mechanism is much simpler.

The provision for this mechanism in IBM FileNet P8 4.0 and above is delivered by a custom JDBC Driver class: `com.filenet.api.jdbc.Driver`. This driver class uses the underlying Content Engine 4.0 Java API to connect to the Content Engine and perform the necessary queries, which means that you have the choice of using the EJB transport or the Web services transport.

The provider URL is specified using the form:

```
jdbc:filenetp8:http://ceserver:9080/wsi/FNCEWS40DIME/
```

In the previous example, `filenetp8` is the driver type that the driver class recognizes, and the URL is the Content Engine connection URL. Extra parameters can be passed or specified in a Java Properties object on create, which are:

- ▶ URI: The Content Engine connection URI.
- ▶ User name: The user name to pass to the JAAS subject.
- ▶ Password: The password to pass to the JAAS subject.
- ▶ JAAS Config Name: The JAAS stanza that indicates the JAAS context to reference. If this is not specified, and no JAAS login context is established in the calling application already (to force a check for this, use the exclamation

point “!” as the value for the JAAS config), the default IBM FileNet P8 JAAS stanza is used and the user name and password are passed to the login module that is specified by it.

- ▶ Object Stores: A list of object stores to query.
- ▶ Merge mode: The merge mode to use if multiple object stores are specified, which can be either *union* (default) or *intersection*.
- ▶ Locale: The Locale to use. If not specified, the default JRE locale of the calling application is used.

This provider is fully SQL 92 compliant, but it does not fully comply to the JDBC API. See the ECM Help JavaDoc documentation for more information about this driver class.

This JDBC provider is also used by the IBM FileNet Records Manager reporting mechanism.

8.8.5 Case Objects as historical records

IBM FileNet Business Process Framework uses the concept of a Case object to store information about the activity that is in progress. After a process finishes, there is no historic record that is retained aside from audit logs of the final state of the process. Case objects can be very useful in showing what the final state was of a process.

Case objects also link to many dependent objects, one of which is the “Audit Log Item”. These log items record actions that happened during the life of the case. These items can be actions that are also recorded elsewhere, such as the updating of properties and completing of process tasks. They could also be application-specific log items, such as user comments or recording that a document was attached or detached from a case.

Case objects provide a richer, more contextual type of log than those that are typically provided by the Content Engine and Process Engine's underlying logging and auditing systems. They are also very useful in allowing employees who work on very specific steps within an overall process to see higher-level status. This feature is useful more in a social aspect because employees do “buy in” more to the overall process and their part within it if they can see how others' actions interact with their own to complete the business process.

These Case objects are implemented as custom objects within the Content Engine and are not versionable and so are useful to record a final record of decisions. Typically, a case object contains properties that are relevant to only some steps in the process, which means that the total set of properties provides

a holistic view of each segment of the process that happened, and therefore versioning is not necessary.

8.9 A practical example: Re-insurance placement and litigation

As a practical example, we consider an enterprise content management solution with business process management features, the re-insurance placement and litigation application. Re-insurance encompasses the concept of large organizations that must ensure, for example, five buildings, a fleet of 80 cars, mobile phones, and a variety of other instruments. They use re-insurers as a single point-of-contact for all of their insurance needs. These re-insurance organizations then break down the total requirement into individual parts that they can get other insurance firms to in turn insure. In our example, this can mean that all vehicle insurance is grouped together and sent to three companies to bid on that part of the overall requirement.

These bids are then collected, and a full package cost is calculated with terms and conditions. A markup is added to cover costs and to create profit for the re-insurer, and this final amount and terms are presented to the re-insurance customer.

In our example, the flow of information is:

1. Customer submits a large requirements document to re-insurer.
2. Document is scanned and routed to the account team.
3. The assigned manager receives the information and starts to put together individual elements to solicit bids.
4. These individual forms are submitted to a select few external insurance agents.
5. When all bids come back, or the time to bid is exceeded, they are evaluated by the account manager who chooses a preferred vendor for that element.
6. A total charge amount and legal terms are created and sent to Legal for approval.
7. After they are approved, they are sent to the customer who signs an agreement, or asks for a reassessment of part of it, or abandons the request.

Designing the business process is out of the scope of this book. Instead, we discuss the security issues that it entails:

- ▶ All documents that are related to a particular customer require access by the same account team.
- ▶ All customer documents must be hidden from other account teams.
- ▶ Ensure that the documents do not leave the customer's legal jurisdiction.
- ▶ Prevent important competitive information from being available to some people on the account team.
- ▶ Creating bid elements by the account manager.
- ▶ Tracking the instructions that an account manager sends to insurers, and what we receive back.
- ▶ Ensuring that late bids are not considered.
- ▶ Preventing bid elements from being modified after the total customer charge and legal terms are considered.
- ▶ Recording every quote that we send to a document.
- ▶ Recording all agreement documents signed by a customer.
- ▶ Managing ad-hoc correspondence.

All documents that are related to a particular customer require access by the same account team

A customer folder is created with *access permissions* set such that the account team can view properties and content of any document that is linked to this folder by the *security folder* property. All subfolders are setup to inherit permissions from this folder. The parent folder itself, must not be set to inherit permissions from its higher-level folder to prevent unauthorized access.

We ensure that new documents that are being scanned into or added to the system initiate a Content Engine action handler that sets their security folder attribute to the correct value. We find the appropriate folder by making sure that the *customer folder* is a special class of folder with a required *customer number* attribute.

We can optionally create special business processes or subfolders to give a greater granularity to where documents are filed. If we receive hundreds of documents from this customer that relates to different matters, for example, we might want to file the incoming documents into the appropriate subfolders.

All customer documents must be hidden from other account teams

We must ensure that the *customer folder* does not inherit permissions from its parent folder. To ensure that security cannot be changed, we must remove any *modify permissions* rights from all non-essential users. We can limit this to just the *Account Manager* on the parent folder, for example, but allow no other person to have this right.

We must also ensure that no privileged permissions are assigned to users, which includes the *write any owner* and *privileges write permissions* on the object store. A user with these rights can modify the document owner and through this its security or modify the content, respectively.

We ensure that all customer documents have *null* in the owner field. We use a separate management workflow process to change ownership rather than allow this to occur manually.

Ensure that the documents do not leave the customer's legal jurisdiction

We create a required multi-value property on all customer documents called *applicable jurisdictions*. We create a marking set that consists of markings that restrict access from users in particular locations, which ensures that the document cannot leave the customer's applicable jurisdictions. We deny all rights from any users outside of applicable jurisdictions who use the constraint mask for each marking.

Prevent important competitive information from being available to some people on the account team

Some document properties, in addition to document content, might include competitive information, such as the value of a quote. We might have some people who are on the account team, such as customer service representatives, who should not see this sensitive information.

At the same time, we might have properties on the document that these users should see, which could be so that they can confirm to a customer that a document was indeed received.

We cannot directly secure individual properties on a Content Engine object, but we can secure objects themselves. Thus we can create an object to hold our securable properties and link this to the document by using an object value property. This object can be set up with a *confidential* security marking to deny the user from seeing this information.

We create a class called *secured property document*, which contains a *secured information* object value property that is required to point to one or zero *security showcase document* objects. These security showcase documents are required to have a value that is set for their security classification. As you can see from Figure 8-15, logging in as a user with sufficient clearance means that we can see the object and its metadata.

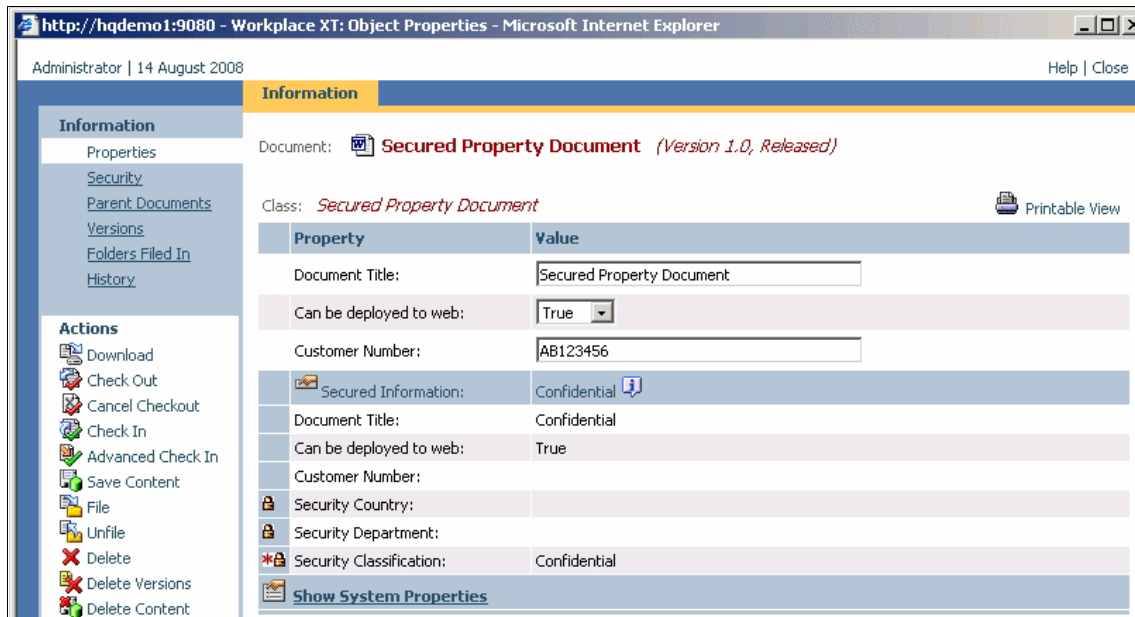


Figure 8-15 Logged in as Administrator, we can see the confidential properties

If we instead login as a user without confidential access, we can read other document properties, but we cannot see the confidential object or its properties, as shown in Figure 8-16 on page 244.

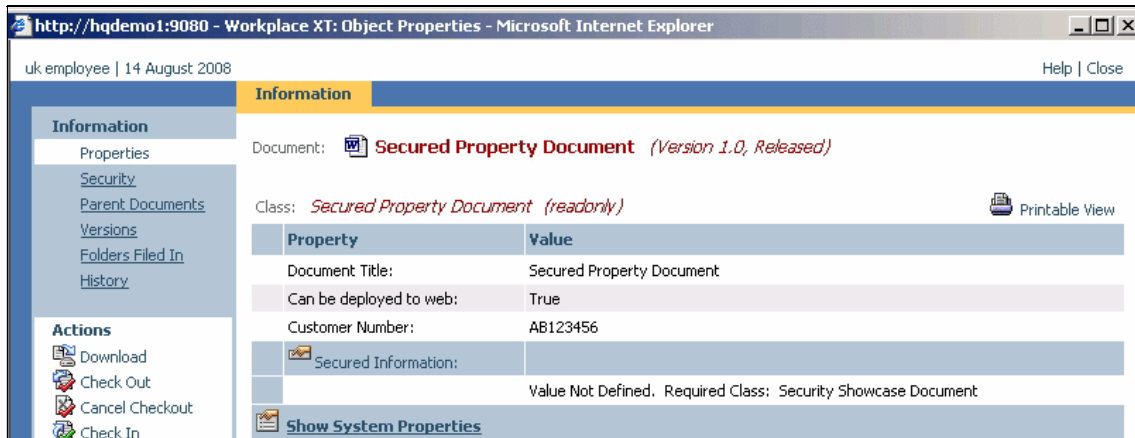


Figure 8-16 Logged in as a user who cannot see the confidential information

We also could define this property as having the document's security inherit from the secured information property and make this property required. If we then set permissions on the confidential information such that view content was denied and inherited by the confidential document and immediate children, and view properties was denied for the confidential document only, there would be no need to explicitly deny view content rights to the top-level document.

In other words, the view content rights would be determined by how sensitive the secured information was. In practice, this is useful if the properties are derived from the main document's content, which means that this security setting can be applied one time on the secured information object and not have to also add a permission to the main document to deny just view content to specific user groups. If confidentiality changed over time, this option is desirable from a manageability perspective.

Note: Do not forget that an explicit direct allow permission on the document overrides an inherited deny for *view content permissions* in this case, which illustrates the desirability of using inherited permissions over direct or default allow permissions.

Creating bid elements by the account manager

We use a business process to create a skeleton structure for a customer deal, which includes a bid elements folder. The Account Manager creates subfolders of class bid element that contains a required bid identifier property. We give the account manager create subfolder rights on the bid elements folder. We also ensure that, by default, the #CREATOR-OWNER of a bid element folder has the file in folder permission.

Because all subfolders of the top level customer folder inherit security permissions and all documents have their security parent property set, we ensure that anyone on the account team can also view properties and content of documents in the bid element folders, which means that the Account Manager does not have to worry about administering security and can instead concentrate on working on the content.

Tracking the instructions that an account manager sends to insurers, and what we receive back

We require an account manager to complete an electronic form with an instruction that is sent to all insurers who are asked to bid on an element. This form drives a business process. The bid element folder is attached to the business process at the same time the form is completed by using a form workflow policy process.

This policy locks down the form so that the account manager cannot modify its content to change what the instruction says or who the bid request is sent to after the form is submitted. To ensure security at the account manager's workstation, we require that the electronic form have a digital signature that the account manager signs just prior to submission. We make this lock down the fields on the form to prevent modification of a signed form by any privileged user.

The business process collects the relevant bid element documents and submits them using an encrypted Web services call to the relevant insurers. Any responses from the insurers are authenticated with the relevant user name and password for this insurance organization to ensure that fake bids do not enter the organization.

Ensuring that late bids are not considered

Upon expiration of the bid timer, the business process locks down the quotes subfolder for the relevant bid element, which prevents an *incoming bid process* from being able to add a bid after the timer is exceeded. If the bid comes later, the incoming bid process can reply to the insurer, notifying them automatically that their bid is not accepted because the submission deadline is not met. It also means that the account team does not worry about whether or not a bid is late. For large accounts, if doing it manually, late bids can be mistakenly considered or modified after submission. By making this electronic and secured, we can prevent this from becoming a problem.

Wait for condition: You can model a *wait for condition* in an IBM FileNet P8 business process to wait for a known maximum number of launched subprocesses, for example, you can have conditions on five routes to launch or wait for a maximum of five sub processes. Use a custom object with an array field called, for example, *SubProcessStates*. In the IBM FileNet P8 process, you launch each sub process in a loop passing it as reference to the custom object instance and the sub process' own index number. Each sub process is responsible for updating its status within this custom object. The main process then periodically checks these values to determine when to continue.

An alternative to periodic checking is to have a Content Engine Java event handler check the custom object every time it is updated. If it finds that all status entries are marked as complete, it can fire a single re-awaken process. The main process then has a wait for condition waiting for this single process, which puts the minimum amount of load on the Content Engine and Process Engine for all of the process synchronization because the wait for condition values are checked when the waited-for process is modified.

Preventing bid elements from being modified after the total customer charge and legal terms are considered

In the re-insurance handling process, we have a system step to add an explicit deny permission for property and content modification to every bid element folder. Because no one can manually add a permission to a bid element document (thanks to default instance security), we know that the modification deny permission at the folder level cannot be overridden by a direct permission on the bid element document because no one can manually add any such permission.

Recording every quote that we send to a document

In case of legal action, we declare all documents that are involved in a re-insurance quote as a critical business record just prior to sending a quote response to a customer. We can use IBM FileNet Records Manager to manage the access to this information so that applying legal holds and managing disposition is automated and centrally controlled according to organization-wide policies, defined by the Legal and the records management teams.

Recording all agreement documents signed by a customer

When a customer agrees with our quote, we must also lock that information down and declare that as part of the same business record. Again, we use IBM FileNet Records Manager to do this.

Managing ad-hoc correspondence

We use the e-mail management features of the IBM FileNet Content Collector to capture all customer correspondence and put them in the relevant correspondence folder. The correspondence is located in our customer folder and is automatically declared as a business record. We can, for example, have a default e-mail storage policy of three years for all customer correspondence. The e-mail can also be declared as a re-insurance quote acceptance record if it is related to a customer re-insurance request, perhaps with a different retention period. IBM FileNet Records Manager resolves these two record types and retains the correspondence accordingly. If bid-related documents are held for five years, for example, this prevents the disposition of the e-mail after the usual minimum of three years, which allows us to prove compliance across different sources of content.

8.10 Summary

In this chapter, we showed that there are many factors to consider when implementing an IBM FileNet P8 solution, which encompasses all elements of security, from authentication and authorization of users to encryption of communications, storage of content, through to proving compliance. Auditing and proving who has or has not modified content is just as important as securing information, and this is increasingly true due to increasing regulation and the litigation nature of business in the current environment.

The IBM FileNet P8 Platform has a rich set of security and auditing features. We showed how these you can leverage these features by expansion products, such as IBM FileNet Business Process Framework, to provide the same capabilities to new business solutions. We also saw how auditing and security management can be extended with other IBM FileNet P8 Platform expansion products, such as IBM FileNet Records Manager.

The IBM FileNet P8 Platform provides a comprehensive set of security features that are sophisticated enough to solve a wide range of business and IT problems. We explained how these features can be applied to new, complex business problems to enable deployments to be conceptualized quickly, which enables customers to rapidly configure, adapt, and extend an IBM FileNet P8 Platform-based solution, while maintaining tight control over the security of the overall system.



Scalability and distribution

In this chapter, we describe the unique ability of the IBM FileNet P8 Platform to scale horizontally and vertically to respond to increasing load demands. We discuss the available options for each of the core platform engines and for the ingestion applications, the application frameworks, and the connectors. We also discuss how a central system can be extended by adding components in satellite locations and the architecture of the system that is used to support a world-wide distribution.

We cover the following topics:

- ▶ 9.1, “Overview” on page 250
- ▶ 9.2, “Scaling the IBM FileNet P8 core engines” on page 257
- ▶ 9.3, “Tuning the IBM FileNet P8 Platform for performance” on page 291
- ▶ 9.4, “Distributing an IBM FileNet P8 system” on page 295
- ▶ 9.5, “IBM FileNet P8 in a DMZ environment” on page 303
- ▶ 9.6, “Sample deployment” on page 306

Disclaimer: In this chapter, we describe a subset of IBM FileNet P8 Platform scalability options and best practices. It is not an exhaustive guide to all options or to every possible install scenario. We recommend that you always include a certified IBM FileNet professional in the process of planning a production architecture.

9.1 Overview

In this section, we briefly discuss the different options that are available to scale IT systems for throughput on an enterprise level. In the subsequent parts of this chapter, we show which concepts can be applied to which component of the IBM FileNet P8 Platform.

Table 9-1 lists some of the terms that we use throughout this chapter.

Table 9-1 Terms and acronyms used in this chapter

Term	Definition
DMZ	Demilitarized zone An area that is normally enclosed by two firewalls to secure internal servers from any harm that originates from potentially insecure external networks that need access to resources in the internal servers
CORBA	Common Object Request Broker Architecture An architecture that the Object Management Group (OMG) manages that allows implementing distributed applications
RMI-IIOP	Remote Method Invocation over Internet Inter ORB Protocol A communication method that CORBA frequently uses
JAAS	Java Authentication and Authorization Service A standard to implement authentication and authorization in the context of a J2EE application.
Content Engine or Process Engine API	The application programming interface that the respective IBM FileNet P8 engines provide.
EJB transport	Enterprise Java Bean transport Communication mechanism that the Content Engine clients use to directly communicate with the Content Engine server EJBs.
WSI transport	Web service interface transport Communication mechanism that the Content Engine clients use to communicate with the Content Engine server through the Web services listener.
WORM	Write Once Read Many Ensures that data written once cannot be altered afterwards.

9.1.1 Horizontal scaling: scale out

Horizontal scaling, also referred to as *scaling out*, indicates that the performance and throughput of the total system can be increased by adding *physical servers*.

Depending on the internal architecture of the application, it might be possible to implement horizontal scaling just by installing the appropriate software components on the additional machine and configuring the application to use them. Commonly this concept is used when the application consists of several components and these components can either be run on a single machine or distributed over several servers.

A different approach to horizontal scaling is the concept of a *server farm*, sometimes referred to as a cluster of nodes, especially in the context of J2EE application servers. The farm consists of multiple instances of a designated server components (such as the Content Engine) that are all identically configured. Although it is possible to configure an application to use one particular server of the farm to get requests processed, in most cases, horizontal scaling introduces a load balancing mechanism, which is used to distribute incoming requests across the servers in the farm for processing. The advantage of this approach is that it can also deliver high availability, if the load balancing device recognizes failed nodes in the farm and excludes them from the load distribution. We continue the discussion of load balancing devices in 9.1.4, “Load balancing” on page 255.

Farming is therefore the preferred approach for scaling layers within the IBM FileNet P8 architecture, which support it because it provides both scalability and high availability.

9.1.2 Vertical scaling: scale up

The concept of vertical scaling, which is also often referred to as *scaling up*, relies on the idea that adding hardware resources, such as memory or CPU power, to a given server results in the ability to handle an increased load.

Depending on the nature of the application that is running on this server, it might be possible that the additional resources are effectively used automatically or just by changing some configuration parameters (for example, the number of threads that are used internally).

In the case of a J2EE application, server infrastructure scaling vertically requires, in most cases, to run multiple instances of the application server on the physical machine to make use of increased hardware resources because each instance of a J2EE server runs in a single Java Virtual Machine (JVM), and the concurrency limitations of a JVM process prevent it from fully using the

processing power of a machine. Also, the amount of time that is required for a garbage collection limits the amount of memory that can be used efficiently by a single JVM. We further elaborate on this topic in 9.3, “Tuning the IBM FileNet P8 Platform for performance” on page 291.

Figure 9-1 illustrates how horizontal and vertical scaling can be used in combination to optimize the usage of resources on a physical server. In this example, on each machine, four instances of the J2EE application server are started and into each application server instance one instance of the application is deployed. As a result, twelve entities of the application are available to handle incoming requests.

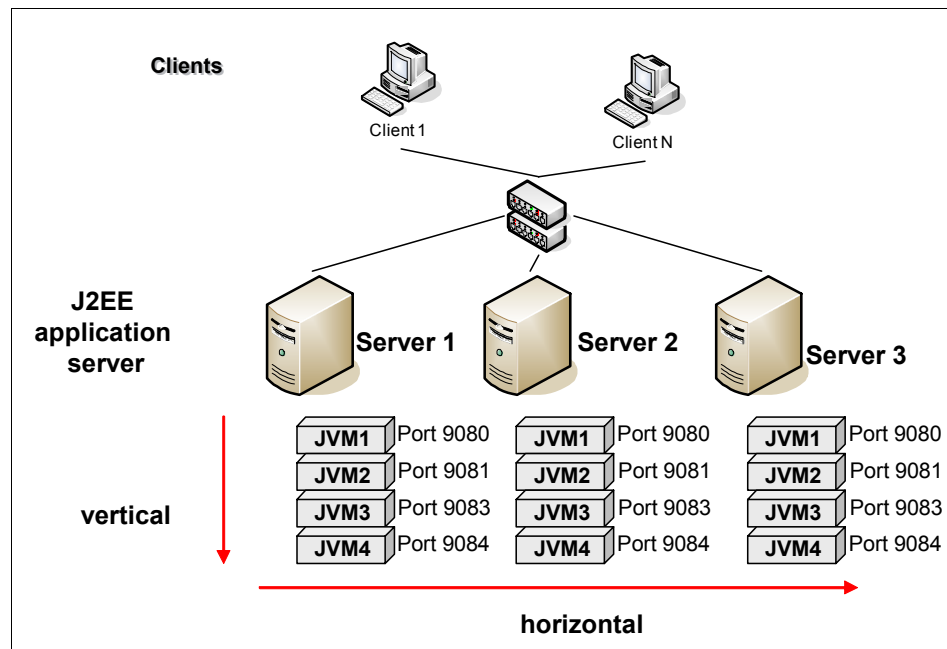


Figure 9-1 Combination of vertical and horizontal scaling for J2EE applications

Certainly, there is a restriction in the number of J2EE application server instances that can run in parallel on a single box without affecting each other in terms of shared system resources, such as CPU, memory, or networking. IBM provides a sizing tool, called *Scout*, to estimate projected system demands based on a given load profile.

Vertical scaling for J2EE applications is an option to reduce the hardware footprint for the solution. By combining horizontal scaling and vertical scaling across three physical machines, as shown in Figure 9-1, high availability can be achieved.

9.1.3 Virtualization

Virtualization describes the concept of partitioning a physical server into multiple logical units that share the hardware resources that the physical host machine provides. Depending on the implementation, the virtual servers either run a completely separate instance of the guest operating system, or they run at least in separated process and memory spaces so that there is no impact on each other.

Virtualization is frequently used to address one of the following challenges:

- ▶ Allow applications that are not supported to run on one server to be co-located on the same physical machine by separating them on individual virtual servers.
- ▶ Implement a sort of vertical scaling for applications that do not natively support vertical scaling and that cannot be installed in parallel into the same instance of the operating system.
- ▶ Improve the flexibility in provisioning hardware resources to applications. Components that run in a virtualized environment can very easily be shifted to other hardware for example.
- ▶ Provide a shared platform build on top of only a few, but very powerful physical servers. Use virtualization to partition the servers into logical units into which the individual applications are installed, thus providing data segregation comparable to a scenario where dedicated physical servers would be used.

Vendors came up with different approaches to provide virtualization. The common aspect is that they introduce an abstraction layer between the device drivers of the operating system and the physical devices themselves, as shown in Figure 9-2 on page 254.

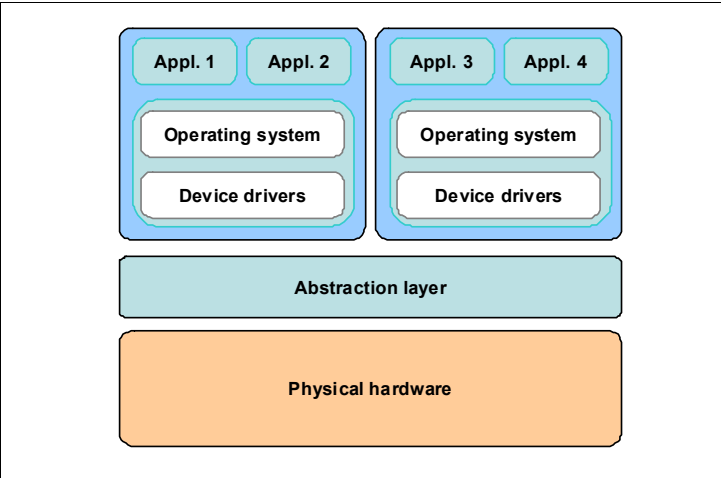


Figure 9-2 Logical architecture of a virtualized environment

In general, virtualization products that are available today fall into one of the three main categories listed in Table 9-2.

Table 9-2 Commonly used approaches to virtualization

Type	Description	Example
Full or native virtualization	Requires a host operating system that provides the abstraction layer. Runs separate instances of the unmodified guest operating systems.	VMWare Server
Paravirtualization	Provides the abstraction layer and has no need for a host operating system. Runs separate instances of a modified guest operating system.	IBM LPAR
Operating system level virtualization	Runs all processes on a single instance of the host operating system and provides abstraction layer and data segregation at the kernel level of the operating system.	Solaris Containers

Be awares

Full or native virtualization is known to have limitations on the performance that the abstraction layer delivers using the host operating system, under certain conditions, for example, in situations of heavy usage of network resources. This heavy use results in negative impact on overall performance delivered by the application that is executed in the virtualized environment; therefore, it is very difficult to provide a proper sizing for this type of virtualization using tools, such as Scout.

Paravirtualization and especially *operating system level virtualization* eliminate the effects of an underlying host operating system and can provide performance that is pretty close to that of the native operating system. Keep in mind that operating system level virtualization might restrict access to certain low-level resources from other operating system entities other than the distinguished master (for example, access to raw devices). Also file systems are shared across all guest operating system entities and access to file systems that are exported by one of the guest operating systems to a guest operating system running on the same machine can be achieved through the loopback device only, not through a regular NFS mount.

Virtualization for an IBM FileNet P8 infrastructure: Not all building blocks of the IBM FileNet P8 portfolio are supported on virtualized hardware, for example IBM FileNet Image Services connected to Jukeboxes. Consult the *IBM FileNet P8 Hardware and Software Requirements Guide*^a for further details when planning to use virtualization for an IBM FileNet P8 infrastructure.

a. http://ftp.software.ibm.com/software/data/cm/filenet/docs/p8doc/40x/p8400_hw_sw_guide.pdf

We recommend using virtualization cautiously, taking into account the potential downsides of the specific type of virtualization that is planned to be used. Virtualization is an option to support the implementation of high availability while coincidentally enforcing consolidation to a few large server systems. Refer to 9.6, “Sample deployment” on page 306, for an example on using virtualization for IBM FileNet P8.

Note: Do not use virtualization in the context of disaster recovery because this purpose requires a separate site.

9.1.4 Load balancing

A load balancing device distributes incoming requests across a number of entities that can service this request. Moreover, the load balancer keeps track of failing entities and prevents sending further requests to them.

Note: If a load balancer can be configured to memorize to which server it forwarded the request of an individual client and to ensure that following requests by this client reach the same server, this is called *session affinity*, *session stickiness* or *session awareness*.

In this section, we describe which implementation of a load balancer can be used for which purpose in the context of the IBM FileNet P8 architecture.

Hardware load balancer

A *hardware load balancer* is an appliance that is specifically designed to distribute a huge number of incoming requests across individual servers (physical or virtual). Typically, the method of distribution can be configured (such as round robin or weighted round robin), and sophisticated units can monitor the back end server load and change the distribution rules accordingly to ensure approximately even loads on the balanced servers. Some hardware load balancers can be extended with additional network ports and provide a huge variety of configuration options, such as considering server processor utilization or the number of connections to an individual server. When building a highly available system, use at least two load balancers to avoid a potential single source of failure because when the hardware load balancer goes out of service, no requests are forwarded to the IBM FileNet P8 servers, and the complete application can no longer be used. At least the load balancer must be hardened by ensuring that critical components, such as network card and power supply, are redundant.

When choosing a hardware load balancer for an IBM FileNet P8 system, it must be guaranteed that it meets at least the following criteria:

- ▶ Support for TCP and UDP
- ▶ Support multiple virtual IP addresses with different rules
- ▶ Support session affinity (sticky sessions)

Software load balancer

A *software load balancer* is an application that runs on a normal server and acts as a proxy that takes incoming requests and routes them to other servers. Similar to a hardware load balancer, the rules for these routes can be configured, which determines the flow of information.

Similar to the hardware load balancer, use at least two installations of a software load balancer to avoid a single point-of-failure in the architecture.

Web server http plug-in

A Web server http plug-in can be seen as a subset of a software load balancer that can only handle http traffic. Such a solution can be used to implement load balancing on Web servers, fronting the actual application servers. In this case, the plug-in configuration determines which incoming URLs are forwarded to which port on which application server. Similar to a hardware or software load balancer, the http plug-in might implement session stickiness and distribute requests from new clients to the application servers that are configured.

Http plug-in-based load balancing is limited to http requests and can handle no other protocols.

Java object load balancing

Routing RMI-IIOP requests across a load balancer can cause undesired results because the request includes the server that is supposed to process this request. This is because per design in CORBA the client itself knows which servers can process a particular request. The client obtains this information during the initial name service lookup using the Java Naming Directory Interface (JNDI) and is responsible on its own to distribute the requests across servers when the JNDI call provides it with a list of servers.

J2EE application servers implement sophisticated load management routines that are applicable to resources, such as Enterprise Java Beans (EJB), across a cluster of servers in different ways, for example, IBM WebSphere Application Server Network Deployment (WAS ND) integrates workload management as a plug-in for the WebSphere object request broker (ORB). The Java Runtime Environment (JRE) that WebSphere provides is required for any remote Java clients to take advantage of the WebSphere load management aware ORB.

9.2 Scaling the IBM FileNet P8 core engines

In this section, we illustrate the options that exist to scale the core engines of the IBM FileNet P8 Platform.

9.2.1 Application Engine

The Application Engine consists of two major parts:

- ▶ The Web application, including:
 - The WebDav listener and
 - The Process Engine XML Web Services listener (P8BPMWSBroker)
- ▶ The Component Manager

Any scaling approach for the Application Engine must take into account both building blocks.

Web application

Technically speaking, the Web application is deployed into the J2EE application server and runs in the Web container of the application server.

The WebDav Listener is implemented as a servlet that accompanies the Web application and runs in the servlet container of the application server.

The functionality that the Process Engine XML Web Service listener delivers is that it:

- ▶ Handles incoming XML Web service messages, which are processed by a Process Engine system step with Receive or Receive and Reply instruction
- ▶ Provides the registry for the Universal Description, Discovery and Integration (UDDI) that external applications can use to find a Process Engine Process that was exposed as an XML Web service for consumption
- ▶ Provides the WSDL lookup to external applications that want to use processes that are exposed as Web services.

The Process Engine XML Web Service listener is also implemented as a servlet. the Component Manager handles outgoing Web Services requests, which we described in “Component Manager” on page 260

We recommend that you establish a server farm to scale J2EE Web applications.

Vertical scaling is also possible by running several instances of the J2EE application server on a single machine, thus deploying multiple instances of the Web application on this server. This approach results in a farm of application server nodes that are, in this case, running on a single physical server.

Note: The approach to distribute the load over the single Application Engine entities remains the same for both horizontal and vertical scaling.

The Application Engine Web application is accessed by the client through a Web browser using the HTTP protocol. These sessions are stateful, thus session affinity (or session stickiness) must be implemented for the load distributing device.

Hardware load balancer

Figure 9-3 on page 259 illustrates using a hardware load balancer to distribute the load across a farm of Application Engine Web applications. It is the most common configuration because such an appliance, in most cases, is already part of the enterprise IT infrastructure and can be used for the IBM FileNet P8 architecture.

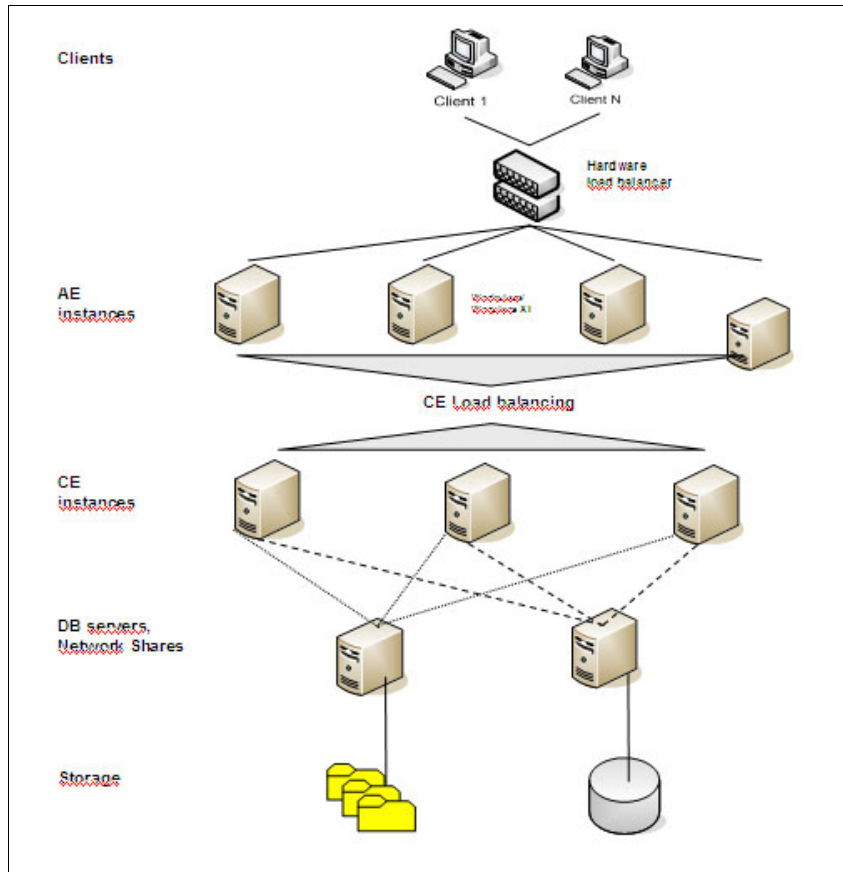


Figure 9-3 Hardware load balancing for Application Engine Web application farm

A common network share must be provided in the farm for all servers that are running the Application Engine Web application. This share is used to store common configuration data. For details, refer to “Installing a Highly Available Application Engine/WorkplaceXT” in the High Availability Tech Note for IBM FileNet P8 4.0:

ftp://ftp.software.ibm.com/software/data/cm/filenet/docs/p8doc/40x/p8_40x_ha_tech_note.pdf

Details about the work load management for accessing the Content Engine are in “Application server load balancing for Content Engine” on page 265.

Http plug-in

When the http proxy servers are securing access to the application layer, an http plug-in can be used to distribute the requests over the Web application farm, as

shown in Figure 9-4. This setup is frequently used in DMZ-style configurations, which we discuss in detail in 9.5, “IBM FileNet P8 in a DMZ environment” on page 303.

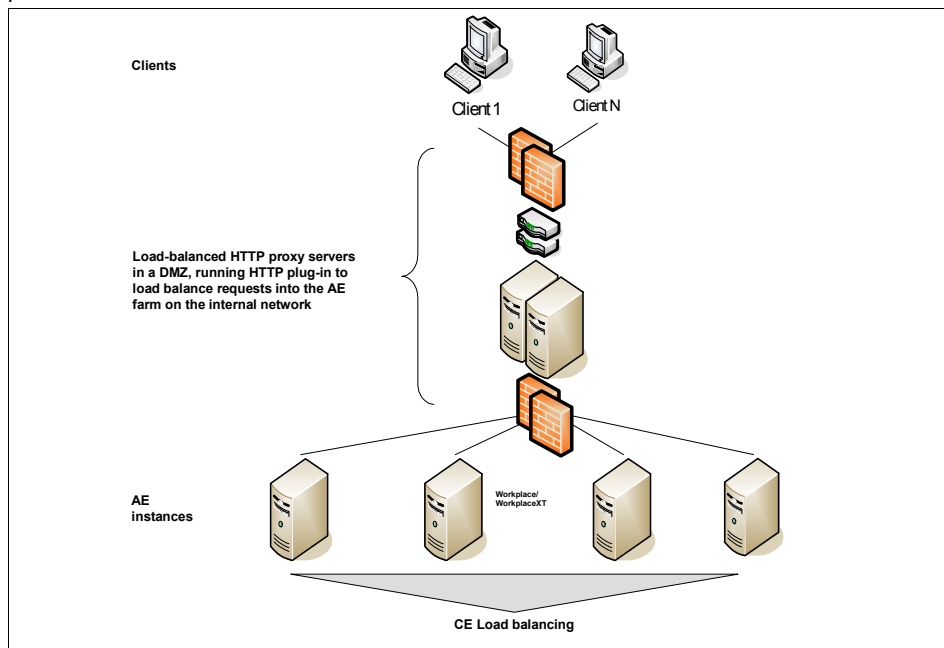


Figure 9-4 Load distribution for Application Engine Web application using an http plug-in

Component Manager

The Component Manager runs standalone in a Java Virtual Machine on the Application Engine server and delivers the following functionality:

- ▶ Dispatches work requests in Component Manager queues to the configured Java classes.
- ▶ Allows interaction with configured Java Messaging System (JMS) queues to write JMS messages from within a workflow.
- ▶ Processes outgoing Web services calls, which are requested by processes that are executed on the Process Engine using system steps with Invoke and Reply instructions.

A single instance of the Component Manager is always bound to one Process Engine-isolated region; therefore, a separate Component Manager instance must be started for each isolated region that the Process Engine uses, in case component integration or process orchestration is used in the processes for that region. Multiple Component Manager instances can be configured using the Process Engine Task Manager.

By default, the Component Manager uses the CE Java API and the WSI transport to connect to the Content Engine and the PE Java API to communicate with the Process Engine. It is supported to configure the Component Manager to use the EJB transport for the CE communication. This is the preferred setup when JMS message queues or if components are used that use client transactions based on the Java Transaction API (JTA) or must get passed the JAAS security context. In this case, we recommend using a separate instance of the Component Manager for each queue using the EJB transport to allow for fine-grained security control.

Using multi threading for components

In the Process Task Manager, you can define for each component how many threads execute this component in parallel. A thread-safe implementation of the component is required if more than one thread is configured. If thread safety cannot be ensured, it is possible to execute another instance of this component in a separate Component Manager entity.

It is also possible to define multiple Component Manager queues for the same Java Component. However, if these queues are configured to be processed by a single instance of the Component Manager, they are all executed as separate threads in the same Java process by the Component Manager. Thus, defining multiple queues for the same component and executing them in a single instance of the Component Manager does not avoid problems regarding thread safety.

Running multiple instances of Component Manager

Each Java component that is configured in the Component Manager is executed in a separate Java process spawned by the Component Manager. The Process Task Manager allows the configuration of multiple Component Manager instances on the same server. For each instance, it can be configured which component queues that processes, which is the preferred approach if components are not thread safe and must be executed in separate Java processes.

Scaling Web service call requests by processes

If process orchestration is used heavily, running multiple instances of the Component Manager allows you to scale the throughput for outgoing Web service calls that active process instances initiate. Each Component Manager instance can be configured to process the outgoing Web services queue (WSRequest). If required, different filter patterns can be defined, for example to ensure that high-priority Web service requests are exclusively processed by certain Component Manager instances.

Because incoming Web service requests for processes are managed by the P8BPMWSBroker, which is part of the Web application (Workplace or

WorkplaceXT), the throughput for processing incoming Web service requests can be increased by farming the Web application.

In both cases, verify that after farming the handlers for increased incoming and outgoing Web service requests, the Process Engine itself can handle the additional load. You can verify using the Scout sizing tool with a baseline that represents the currently existing system load.

Workload distribution

In general, the Component Manager instances poll their work from the configured component queues rather than getting requests pushed out. There is a single exception from this rule because the Process Engine can notify a single Component Manager instance that new work arrived using the Component Manager event port. This feature can be used to configure a large polling interval for the Component Manager and ensures that new work items are nevertheless processed quickly by the Process Engine notifying the Component Manager. In terms of scaling, it is not required to implement a load management for this notification. It is more efficient to use a smaller polling interval if a large amount of work for the Component Manager is expected, which must be the reason why this component is supposed to be scaled; otherwise, Component Manager instances can run in parallel to increase the number of requests that are processed.

The Component Manager configuration includes the link to the Application Engine Web application, which communicates with the Process Engine XML Web Services listener servlet. Therefore, in a farmed environment, the virtual address of the load balancer or the http proxy must be configured for the Component Manager instances.

9.2.2 Content Engine

The Content Engine is a J2EE implementation. Therefore, Content Engine supports the standard approach for scaling J2EE applications by establishing a farm of nodes across which the load can be distributed.

Additionally, the Content Engine hosts two Web service listeners: for the Content Engine (CEWS) and for the Process Engine (PEWS). Both listeners are the entry points that clients can use to communicate with either Content or Process Engine using their Web services API and are running in the Web container of the application server.

Horizontal scaling is the preferred option for applications that run in the context of a J2EE application server because it also provides high availability at no additional cost. Horizontal scaling establishes a farm of Content Engine nodes.

If vertical scaling is preferred, it is implemented by multiple instances of the Content Engine into multiple instances of the application server, running on the same physical machine for the Application Engine. To obtain high availability, we recommend that you scale vertically using at least two physical machines.

Content Federation Services (CFS)

Content Engine does not support the division of particular components or services to a separate server. Instead, each instance of Content Engine provides the full set of functions, which eases up the scaling. Thus, Content Federation Services (CFS), which is an integral part of the Content Engine, can easily be scaled by adding Content Engine servers to a given system.

No load balancer fronting the Content Engine

In general, Content Engine can be scaled without the need for a load balancing device fronting the Content Engine servers. In this case, the load distribution must be implemented on the layer of the application(s) that use the Content Engine servers, as illustrated in Figure 9-5 on page 264, where three different applications (Appl. 1, Appl.2, and Appl. 3) are used. These applications are running on the Application Engine servers, and the Content Engine APIs on the application servers are explicitly configured to use a particular Content Engine server. For Appl. 3, a farm that consists of two Application Engine servers is used and both are configured to work with the same Content Engine server.

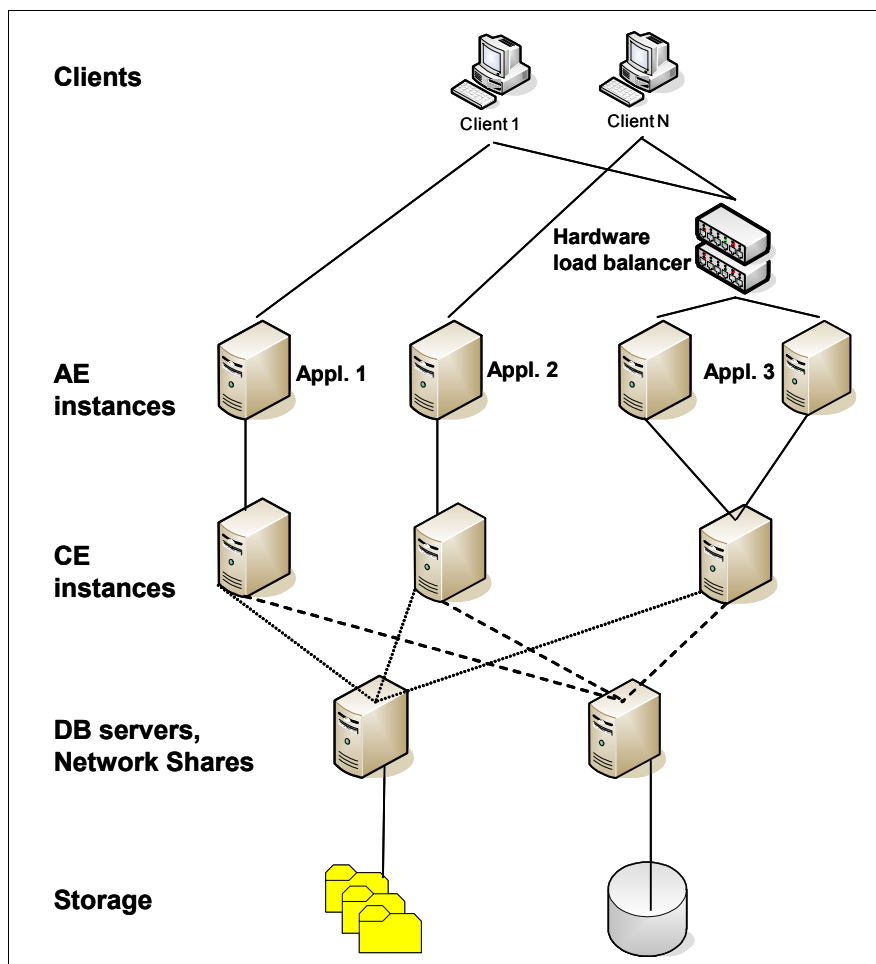


Figure 9-5 Configuration where applications use a particular Content Engine server

The approach in Figure 9-5 requires sizing the Content Engines explicitly for each application, and it has the drawback that, for example, a failure of left most Content Engine server stops Appl. 1 from working at all because no load distribution across the boundaries of an application happens. The advantage of this approach is that the Content Engine server resource is dedicated to a certain application so that the failure does not impact the other applications. Also, with one application, for example Appl. 2 faces severe load whereas the others do not, there is no way that Appl. 2 can benefit from the capacities that are available from the two other Content Engine servers.

For this configuration, no restriction applies regarding the protocol that can be used by the applications to access the Content Engine.

We do not recommend a configuration of this type because it does not provide flexibility to provision the processing power, which is supplied by the Content Engines that are installed to the different applications running. In addition, high availability is not addressed by this configuration.

Application server load balancing for Content Engine

If the applications that access a Content Engine farm use the Content Engine Java API and the EJB transport (such as Workplace, WorkplaceXT, Records Manager, BPF or any custom Java application), and it is not desired to achieve the load balancing on the level of the application, which we discussed in the “No load balancer fronting the Content Engine” on page 263, load distribution must be implemented on the level of the application server into which the Content Engine is deployed. In this case, the load management for EJBs that is built-in into the application server is used to distribute incoming requests to the Content Engine EJBs via the Java API.

All application servers that are supported to run the Content Engine offer support for EJB load balancing although the concepts vary in detail. We illustrate this approach for the example of IBM WebSphere application server network deployment (WAS ND).

At the level of WAS ND, a logical unit called a cluster is defined and Content Engine instances are assigned to this cluster. A Content Engine instance refers to a single deployment of the Content Engine into a single application server instance of WAS ND. For vertical scaling, several application server instances of WAS ND might run on a single physical server using different ports, whereas for horizontal scaling only a single port per server is required.

The applications that use the Content Engine must be configured to use the cluster address instead of an individual server address to take advantage of the work load management that is built into WAS ND by properly configuring the URL that points the Java API to the Content Engine.

Figure 9-6 on page 266 illustrates a configuration with load management for the Content Engine that the application server provides. For simplicity, the connections for the application servers are drawn explicitly.

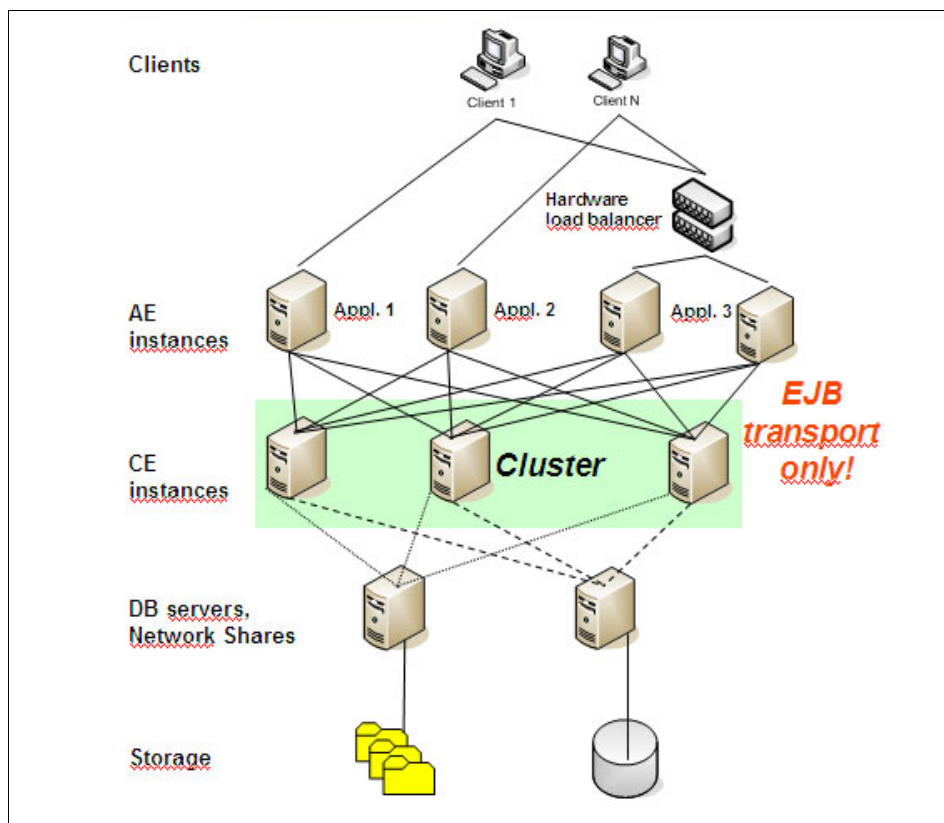


Figure 9-6 : Application server cluster based load balancing for Content Engine

The configuration in Figure 9-6 is probably the most common for IBM FileNet P8 architectures for clients using the Java API because it provides scaling and high availability for the Content Engine. Additionally, when used with a product, such as WAS ND, the deployment process for the Content Engine over the farm can be sped up significantly because the Content Engine must be deployed only one time to the reference node, and the deployment manager then updates the other nodes in the cluster accordingly.

In Figure 9-6, the AE instances, DB servers, and network shares are shown as singletons, which is not the case in a true high-availability deployment. System-wide high availability requires redundancy across the board.

Note: Application server-based load balancing for the Content Engine only addresses clients who use the CE Java API and the EJB transport. Additionally, you must provide either hardware or software-based load balancing for any other clients using, for example, the WSI transport, such as the Component Manager or the Process Engine.

Hardware load balancer

If a hardware load balancer is supposed to be used for balancing the load across the Content Engine servers, configuring the Content Engine API to use the Web Services transport (WSI) for each client that accesses the Content Engine using the hardware load balancer is required. Therefore, this method is not applicable for:

- ▶ Workplace or WorkplaceXT
- ▶ Records Manager
- ▶ Business Process Framework

For all of the applications that we previously mentioned, you must use application server-based load balancing. Custom Java applications might use the WSI transport for the Java API if they do not use features, such as client-based transactions, which require the Java API to use the EJB transport.

Figure 9-7 on page 268 shows a hardware load balancer fronting a Content Engine farm.

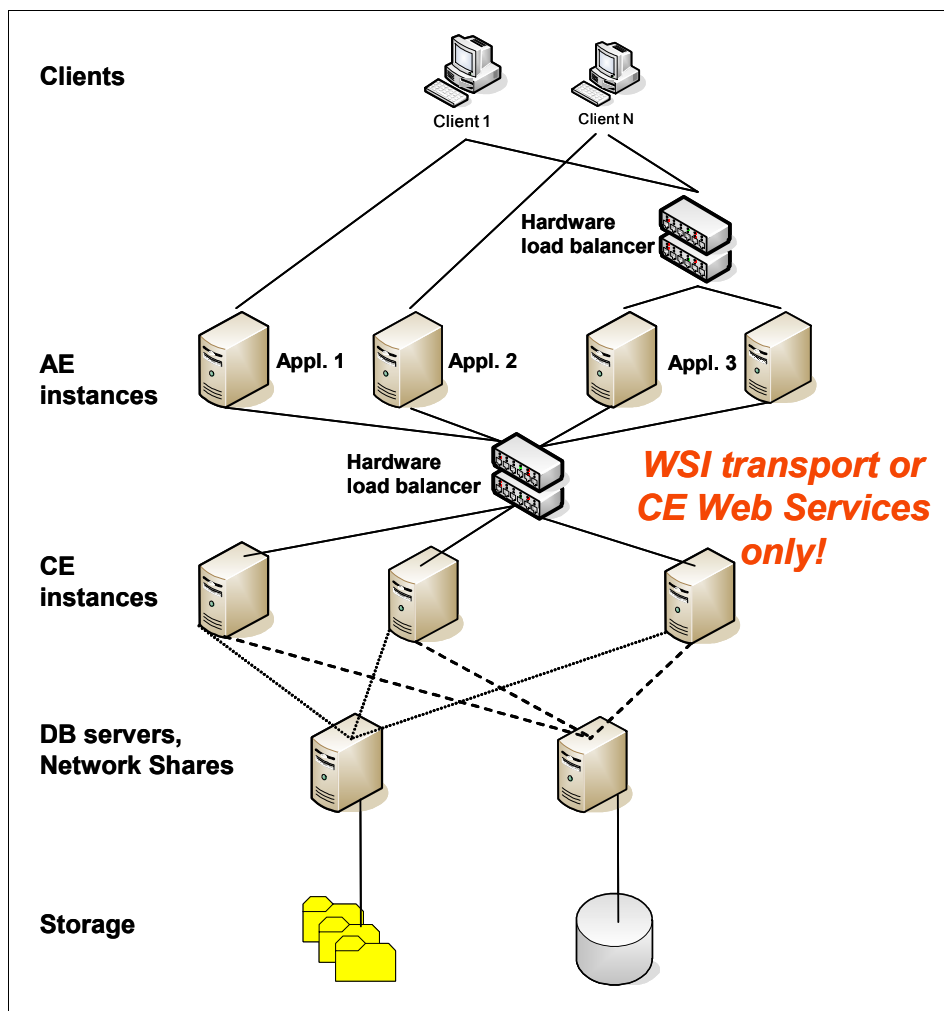


Figure 9-7 Hardware load balancer fronting a Content Engine farm

Applications that use the Web Services API to access the Content Engine (such as any .NET based application) can also take advantage of a hardware load balancer.

From a configuration point-of-view, it is only required to use the virtual host name that the load balancer provides in the configuration of a Content Engine Web Services API on the client. Because the communication between a client and the Content Engine is stateless, it is not required to configure session affinity on the load balancer.

Figure 9-7 on page 268 illustrates a configuration that uses a hardware load balancer. It is not necessarily required to use separate load balancers for the application layer and the Content Engine layer. It depends on the actual environment whether the same load balancers are used for both purposes. If the application layer is located in a DMZ, it is less likely that the hardware load balancers can be shared for the Content Engine load distribution.

Software load balancer or http plug-in

Similar to a hardware load balancer, a software load balancer or an http plug-in can be used to distribute the traffic from clients using the Content Engine Web Services or the WSI transport when communicating with the Content Engine. This configuration requires an http server in the communication path between the client application and the Content Engine. A typical use case is a design where a Web server in the DMZ acts as a proxy and forwards the requests to the actual application servers. It is less likely to install additional Web servers or a software load balancer in a layer between the application servers and the Content Engine servers to obtain load balancing, although this is possible.

Content storage

Table 9-3 shows the options for storing content elements with the Content Engine. See 2.2.6, “Storage services” on page 34, for more information.

Table 9-3 Storage options for Content Engine

Storage area	Description
File storage area	Content stored in a folder hierarchy on a shared file system
Database storage area	Content stored as binary object (BLOB) in the object store database
Fixed storage area	Content stored on a supported fixed content device (FileNet Image Services, IBM N-Series with SnapLock, EMC Centera)

The storage subsystem becomes the bottleneck for the throughput if it fails to deliver the ingestion or retrieval rates that the rest of the architecture could accomplish, such as the database for metadata storage.

File storage area

Consider that the file storage area is the normal way that the Content Engine stores content elements, because:

- The cost per GB is lowest for this type of storage compared to database storage or a fixed content device.

- ▶ File storage areas can be farmed to spread the creation of content objects across multiple storage areas by using a storage policy that maps to 1-N different storage areas. Doing so reduces the risk of congesting a storage area's inbound folder into which each content element is placed prior to being moved into the final location if a large number of objects will be ingested. For details about this process, refer to *IBM FileNet Content Manager Implementation Best Practices and Recommendations*, SG24-7547. Farming the storage areas helps to better utilize and tune the resources for storing the content in a file store.
- ▶ On the level of the file shares, the underlying servers can be tuned accordingly to properly handle the incoming files and the network bandwidth for sharing the file system.
- ▶ At the network level, it can be ensured that sufficient bandwidth is available for the Content Engine servers to write the files to the network shares. Additional network cards can be added if required to enhance the bandwidth.

Therefore, we recommend using file storage areas in general. The database or fixed content storage area options must only be considered to satisfy a special use case.

Database storage area

Using a database storage area requires that all content is transferred into a *single* object store database. Whereas it is possible to scale the uploading units by adding Content Engine servers, all content elements must be received and stored by the database engine. This traffic uses the same infrastructure components of the J2EE server, such as the metadata, for example the JDBC connection pool to the database. Compared to the file store example that we previously discussed, there are fewer options to establish parallel communication channels. Thus scaling relies heavily on the capability of the underlying database system.

A database storage area delivers the benefit that content and metadata is stored in a single database, which makes backup and restore scenarios easier because there is no need to ensure synchronization between a file system or fixed content device and the metadata database.

Fixed storage area

Fixed storage areas are often used if the system must fulfill certain compliance requirements. In many cases, it is easier to ensure, on the level of the storage device, that content that is written once cannot be changed later on for a defined period of time (referred to as the *retention period*). Based on the concept of the content life cycle (refer to 7.3.2, "Life cycle and content storage" on page 164), the Content Engine allows the content to be moved to such a device, for example when a status is reached where content is archived for a certain retention period.

Thus, fixed content devices do not primarily compete with file storage areas regarding the maximum ingestion speed, but it comes in handy if WORM storage is a requirement.

Fixed storage areas basically show the same pattern as a database storage area because all content elements typically get stored on a single fixed content device. In contrast to the database storage area, there are two separate channels that are used for storing the metadata (in the database) and the content (on the fixed content device). Alternately, the speed for storage and retrieval operations and the throughput that is achievable varies between the different fixed content devices that the Content Engine supports. Generally, the performance of a fixed storage area is inferior to a file or database storage area.

For the fixed storage areas, a *staging area* on a network share must exist. New content elements are placed into this staging area before they are moved to their final destination on the fixed content device. For high ingestion rates, you must farm multiple fixed storage areas using a storage policy, although this does not change the fact that a single fixed-content device finally stores all of the content elements; however, it helps to ensure that the staging areas do not become congested.

Benchmark

In various benchmarks, the Content Engine delivers superior performance and scales extraordinarily well. The latest benchmark proved that the Content Engine showed near-linear growth in throughput when additional instances were added to the Content Engine farm. The system consisted of up to 16 Content Engine instances deployed into WebSphere Application Server instances. Refer to the white paper *IBM FileNet P8 4.0: Content Engine Performance and Scalability*¹ for details.

The tests were conducted with a single object store and a farm of file storage areas. An IBM FileNet P8 system at the enterprise level might deliver higher throughput in a real world deployment. An enterprise deployment typically includes more than a single object store. The benchmark implied that the performance of the database will probably become the bottleneck when scaling further. When the ingested content is distributed over several object stores, which can be hosted on different database servers if required, the load pattern for any single database instance drops.

¹ Available on request.

9.2.3 Content Search Engine

The Content Search Engine provides the ability for content based retrieval (CBR) on documents, metadata, and annotations (if configured).

Two major functions of the Content Search Engine are:

- ▶ Create fulltext index information
- ▶ Execute searches utilizing the fulltext index

The majority of use cases focus on scaling the fulltext creation because a high throughput is required for this component to make new documents available for fulltext search after their ingestion as quickly as possible. However, use cases also exist where documents are mainly searched by their fulltext information, so rolling out solutions of this type also require that processing the fulltext search requests are scaled too.

Fulltext indexing

The fulltext is maintained in a Verity K2 fulltext engine. The following steps are executed when a CBR-enabled object gets created or updated:

1. The Content Engine server creates a row in the IndexRequest table.
2. A background task, the CBR Dispatcher, reads a batch of rows from the IndexRequest table and hands them over to a CBR Executor.
3. The CBR Executor gets the information for the batch and submits it to a Verity “Index Server”. For content in a file store area, the location of the content object is handed over. For content in a database storage area, the content object is pulled out of the database, written to a temporary file, and then handed over to the Verity K2 Indexer process.
4. The Verity K2 Index Server writes data to a *Verity Collection*, which is similar to a database table and used by Verity K2 internally to maintain the fulltext information.

The indexing process executes asynchronously on the Content Engine.

Verity collections: Verity collections have capacity limits, so many of them must be created in a large system.

Scaling out indexing

Verity K2 allows you to configure multiple K2 servers to work as a cluster to provide high availability and scaling. This configuration improves the throughput for indexing requests at the level of Verity K2 because multiple indexer processes can work in parallel on the different nodes of the Verity K2 cluster.

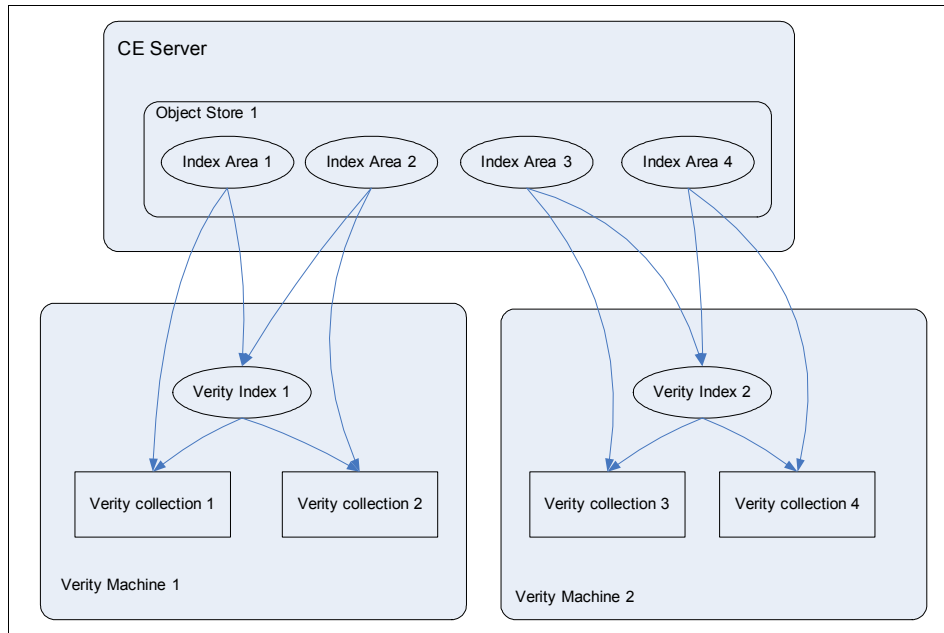


Figure 9-8 Scaling out Content Search Engine indexing

Each server in the Verity K2 cluster is configured to handle one or more *Index Areas*. Using at least two Index Areas on each Verity K2 server improves the throughput due to an increase in concurrent writes. Figure 9-8 illustrates a configuration with multiple Verity K2 servers and index areas. Each search server handles two search areas in this example.

The Content Engine CBR Executor hands over an indexing request to any of the servers that are configured for the appropriate Index Area in a random fashion, thus distributing the load.

Note: Because of a limitation in K2 Verity, currently only one CBR dispatcher process can be active for an IBM FileNet Content Manager site. Content Manager V4.5 provides a new feature that ensures that for each site, only one Content Engine server runs the CBR dispatcher process, and if one is down, it automatically launches a dispatcher from another server to perform the task.

Fulltext search

Similar to the indexing phase, any content-based search on document, metadata, or annotation content involves a communication between the Content Engine and the Verity K2 fulltext engine.

The following steps are executed:

1. The full text part of the search request that is submitted to the Content Engine is extracted and handed over to a *Verity K2 Broker* process.
2. The Verity K2 Broker calls a Verity Server that searches the full text data.
3. The resulting rows from the Verity K2 Broker are stored in the temp table of the Object Store database.
4. A relational query is executed that joins the temp table with the remainder of the query submitted. This relational query returns a subset of the rows that are returned from the full text search.
5. The resulting rows are passed back.

The fulltext search is executed synchronously on the Content Engine.

Scaling out fulltext search

To raise the number of search requests that can be handled simultaneously, the number of Verity K2 search servers must be increased. If multiple Verity K2 search servers are available, the Verity K2 broker process issues the incoming requests to all search servers simultaneously where they get processed concurrently. The Verity K2 broker gathers the results that the search servers deliver and returns them to the Content Engine.

Figure 9-9 on page 275 illustrates the use of multiple Verity K2 search servers to scale out content-based retrievals. There is only one Verity K2 broker process required (on Verity server 1) that accepts the search requests that the Content Engine executes and dispatches them to the search servers.

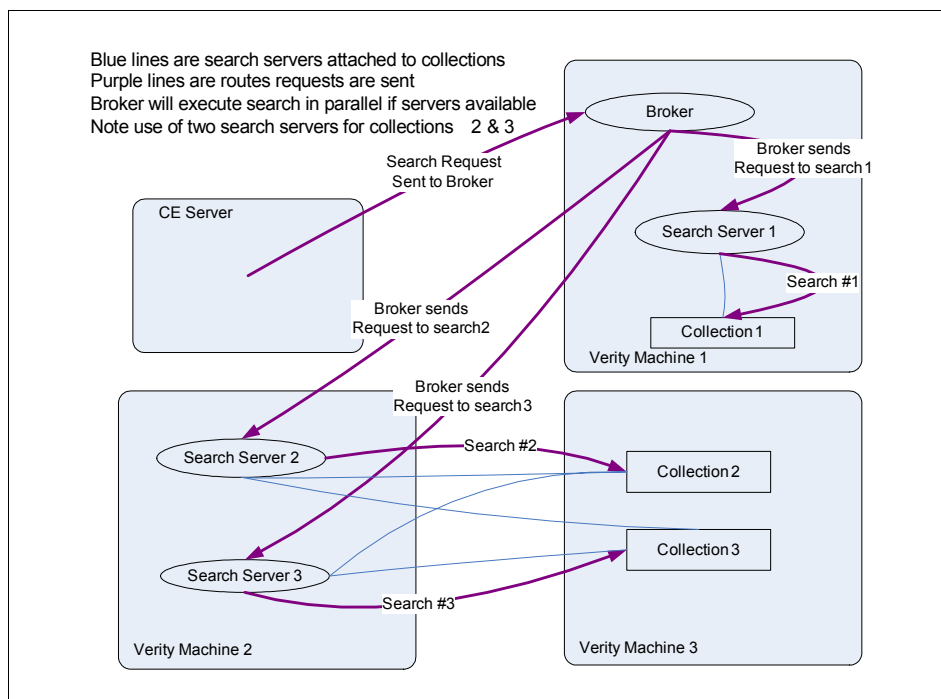


Figure 9-9 Scaling out Content Search Engine retrievals

We recommend using advanced technologies, such as the classification module, to extract useful metadata from document fulltext instead of purely relying on a fulltext search engine. The availability of such metadata is key in an Enterprise Content Management strategy because it is the fastest way to make content accessible for a wider audience of users. Being able to use metadata, sometimes in conjunction with fulltext information, is the superior concept of relying on the fulltext information as the primary source for finding content.

9.2.4 Process Engine

The Process Engine is a C application, so it does not run in the context of a J2EE application server. Nevertheless it supports both horizontal and vertical scaling to respond to increasing system demands. In this section, we highlight the different options that are available based on the Process Engine internal architecture, which we described in detail in 2.3, “Process Engine” on page 40:

- ▶ Single Process Engine server
- ▶ Farming Process Engine servers
- ▶ Independent Process Engine servers

Single Process Engine server

If the Process Engine system consists only of a single server, scaling vertically is the only available option. The Process Engine functions are executed by worker processes that are controlled by a central broker instance. This broker starts the worker processes that dispatch the process instances (*work objects*) and the background processes that manage the E-mail notification and similar tasks.

If additional resources in CPU power and memory are available, it is possible to increase the number of parallel processes, which enables the Process Engine server to handle an increasing number of work objects.

Farming Process Engine servers

The Process Engine supports farming of Process Engine servers. In such a configuration, multiple Process Engine servers access a shared database that stores the information, such as work objects and queues.

Figure 9-10 on page 277 shows a configuration where two separate applications access a shared Process Engine farm, which is a common configuration because both applications can use different isolated regions, which ensures that work objects do not interfere.

Farming Process Engine Servers is introduced with IBM FileNet P8 4.0 and allows customers to implement the same concept of scaling horizontally across all core engines of the Platform. In addition, it provides high availability without the need for spare servers running idle.

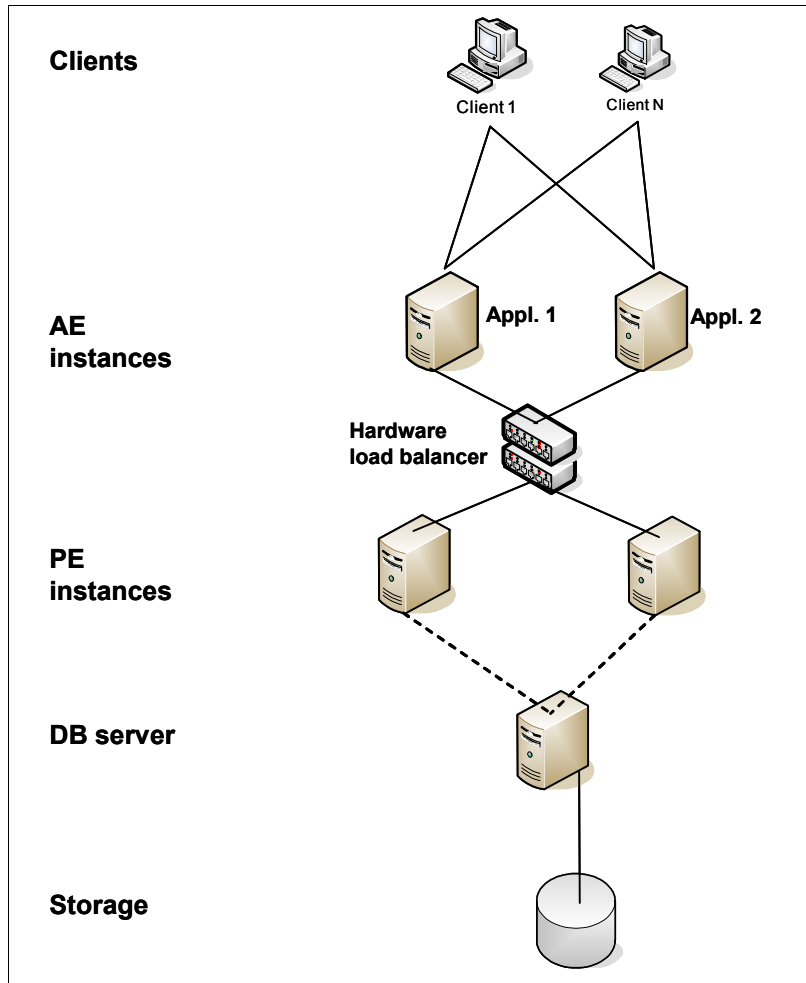


Figure 9-10 Process Engine system with server farm

Farming the Process Engine servers requires a hardware load balancer to distribute the incoming requests. Because the connections are stateless, session affinity is not required. Details about the installation and configuration are in the IBM FileNet P8 High Availability Technical Notice.

Using the Process Engine Task Manager, all servers in the Process Engine system can be managed from a central location. This central management includes starting and stopping the Process Engine software on single nodes and removing nodes from or adding nodes to a farm.

Farming is also the preferred way to scale the Process Engine. This concept delivers high availability because the remaining servers continue to deliver the Process Engine functionality if a single node in the farm fails.

IBM conducted benchmark tests that proved a near-linear growth on transactions handled by a Process Engine Farm when additional servers were added to the farm. Because the Process Engine scales extremely well, the database that is the Process Engine farm uses might become the bottleneck when scaling horizontally.

Independent Process Engine servers

The Process Engine system can also be scaled by using independent servers. As Figure 9-11 on page 279 illustrates, each Process Engine server uses its own database, which can be hosted on an individual database server.

Scaling with independent servers is an option to scale when the overall architecture is geographically distributed and work is mainly done locally, for example, the server for application 1 can be in Los Angeles and the left Process Engine server and the left database server, whereas the servers on the right are located in New York to support another application that is used there. The fact that users in each location always work with local servers and a local database delivers optimal performance.

The drawback of using independent servers is that they do not deliver high availability. If one server fails, the isolated regions that are hosted on this server are no longer available.

Two alternatives exist regarding the Content Engine for storing the process-related content objects. Both applications can use a shared Content Engine server farm, or both applications can use separate Content Engine servers. In the example of the distributed environment, separate Content Engine servers are configured, one for each location to ensure that the application can locally retrieve the content.

It is possible to mix both options when designing an IBM FileNet P8 system. Referring to the previous example, and assuming that high availability is required only for the application in New York. This results in a configuration where a Process Engine farm is configured for New York, and an independent Process Engine server is installed in Los Angeles.

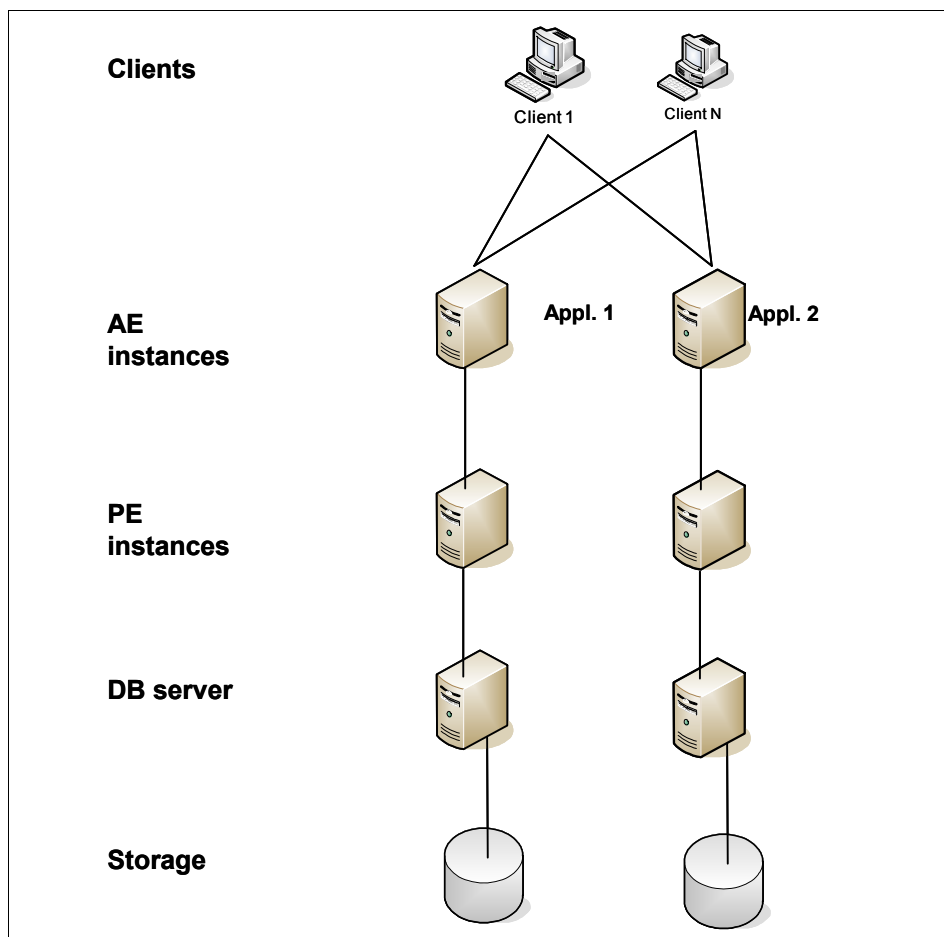


Figure 9-11 Process Engine system with standalone servers

9.2.5 Summary

The IBM FileNet P8 architecture supports horizontal and vertical scaling for all core platform components to respond to increasing system demand. Benchmarks show that IBM FileNet P8 shows nearly-linear scaling over a wide range for both the Content and Process Engine.

Using the approach of farming the layers for the Application Engine, Content Engine, and Process Engine, IBM FileNet P8 provides a solution that makes it very easy to add resources to a given system. The only thing that you must do is provision additional servers (or additional instances on existing servers, where

applicable) with the corresponding platform server component, and reconfigure the workload management component to actively use this component.

9.2.6 Scaling add-on products

There are many products that are IBM FileNet P8 based. In this section, we discuss scaling of some of these add-on (expansion) products:

- ▶ Business Process Framework
- ▶ IBM FileNet eForms
- ▶ Application Connector for SAP
- ▶ IBM FileNet Capture
- ▶ IBM Content Collector
- ▶ Connectors for Quickr and SharePoint
- ▶ Process Analyzer
- ▶ Process Simulator
- ▶ IBM FileNet Business Activity Monitor

For an introduction to these add-on products, refer to Chapter 2, “Core component architecture” on page 27.

Business Process Framework

The Business Process Framework (BPF) can scale in the same way that the Application Engine scales because it has a similar structure. The core components of BPF are:

- ▶ The Web application that is deployed on a J2EE server
- ▶ A Component Manager queue for BPF-related operations that can be triggered by a process executing on the Process Engine

We discussed all scaling options for the Application Engine in 9.2.1, “Application Engine” on page 257 also apply for BPF. Similarly, we recommend farming the Web application to gain benefit from the high availability, which is automatically introduced by this architecture. For the Component Manager that hosts the BPF operations, the best practice is to configure additional instances to handle an increasing number of requests. However, because the BPF operations components are implemented thread safe, it is also possible to configure multiple threads for a single BPF operations queue.

It is important to remember that one BPF application, just like Workplace or WorkplaceXT, can only be configured to use one isolated region. This is no restriction in flexibility, and for the sake of data segregation work objects cannot cross the border between isolated regions. Therefore, it makes sense to implement individual BPF applications for each isolated region.

IBM FileNet eForms

Electronic Forms for IBM FileNet P8 are basically an extension to an existing Workplace or WorkplaceXT installation. As such, IBM FileNet P8 eForms can be scaled in the same way that the Application Engine Web application is scaled. Because eForms are an extension for Workplace or WorkplaceXT, scaling the Application Engine automatically takes care of eForms because this product is installed on the Application Engine.

eForms provide several options to integrate with external systems, for example: databases by taking advantage of JDBC lookups or arbitrary systems by using HTTP calls to lookup data. You must ensure that the system that eForms integrates with and the intermediate piece that facilitates this communication (for example, a servlet performing a lookup against a host system) are designed accordingly so that they can handle the increased load that originates from scaling the eForms and the Application Engine.

Application Connector for SAP

The IBM FileNet P8 architecture embraces two different connector products for SAP systems:

- ▶ Application Connector for SAP R/3 (ACSAP R/3)
- ▶ Application Connector for SAP Enterprise Portal Knowledge Management (ACSAP EP-KM)

Because both products are addressing different use cases, we separately discuss the options regarding scaling.

ACSAP for R/3

The ACSAP for R/3 is used in the context of the classic SAP R/3 Enterprise Resource Planning (ERP) application and adds the following capabilities:

- ▶ Allows users to access documents that are linked to SAP objects and that are stored outside of SAP in an ECM repository.
- ▶ Stores outgoing documents (for example invoices and print lists) that SAP R/3 created in an ECM repository and links them to SAP objects.
- ▶ Archives SAP data that is no longer used online. In this case, contents of the internal SAP database tables are exported as binary blobs and stored in the ECM repository. By request, this data can be re-imported to make it available online again.

The first use case has two aspects: Creating the linkage between an SAP object and a document in the repository, which is typically a batch process done in the background and the ad hoc retrieval of a linked document from the SAP GUI by the end user. The content is the size of a typical document.

The second use case is typically run as a batch job when SAP spools newly created outgoing documents to a shared device where an ACSAP component picks them up, stores them in the repository, and delivers back the reference to the object. Outgoing documents can vary in size because they can be normal documents (an invoice) or long lists (account statements) that can have a size of hundreds of megabytes.

In the third use case, SAP hands over large binary objects that contain exported archived table contents to the ACSAP component for archival purposes. Again, the objects to be stored are large (hundreds of megabytes).

ACSAP for R/3 is a Web application that will be deployed into a J2EE application server. Therefore, the options that we previously discussed for the Web application of the Application Engine, Workplace, or WorkplaceXT apply also for ACSAP, which is vertical and horizontal scaling. Again, in both cases, multiple instances of ACSAP are deployed into multiple instances of the application server (refer to Figure 9-1 on page 252 for more details).

Because ACSAP for R/3 is integrated with SAP, there are configurations that are stored in SAP R/3 that enable the SAP system to determine which server to contact for archival requests (store and retrieve). If only a single SAP instance exists, scaling ACSAP for R/3 is typically addressed by establishing a farm of ACSAP instances, fronted by either a hardware or a software load balancer (as discussed for other Web applications before); therefore, you can configure a single virtual connection information in SAP that points to the load balancer and is distributed across the farm of ACSAP instances, as shown in Figure 9-12 on page 283. The ACSAP instances can either be deployed in separate instances of a J2EE application server on the same server (vertical scaling) or on separate servers (horizontal scaling).

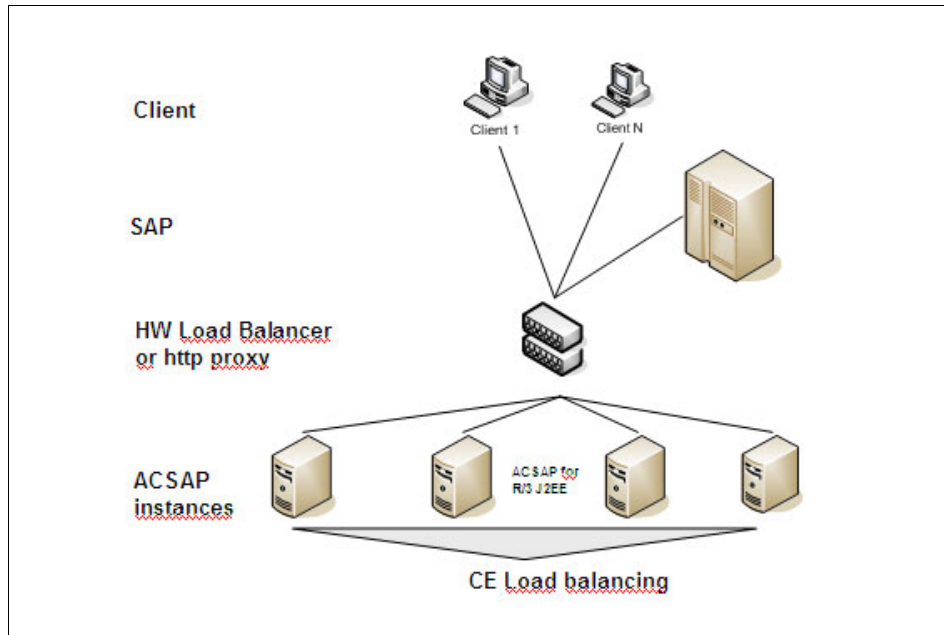


Figure 9-12 Scaling ACSAP for R/3 with IBM FileNet P8 Content Engine

Because the current implementation of ACSAP uses the Content Engine Java API, load balancing the connection between the ACSAP instances and Content Engine requires J2EE server-based load balancing, if the EJB transport is used (refer to “Application server load balancing for Content Engine” on page 265 for more details).

If several different SAP R/3 systems are in use, which is a common pattern in the customer base, typically dedicated ACSAP installations serve each SAP R/3 system. In this case, either individual ACSAP instances are configured for an SAP R/3 system or smaller farms for ACSAP are established if the load is larger than a single ACSAP instance can handle. Using farmed ACSAP instances also provides high availability whereas individual independent ACSAP instances configured for dedicated SAP systems does not.

ACSAP for R/3 and IBM FileNet Image Services

ACSAP for R/3 can also be used in combination with IBM FileNet Image Services as a repository. In this case, the architecture for scaling is nearly identical to the scaling architecture in Figure 9-12. The Image Services Resource Adapter (ISRA) establishes the connection to the repository, which must be deployed in a J2EE application server instance.

Commonly, the J2EE server instance for ACSAP also hosts ISRA. ISRA supports horizontal and vertical scaling by deploying multiple instances, which we previously discussed in the context of Web applications (refer to Figure 9-1 on page 252).

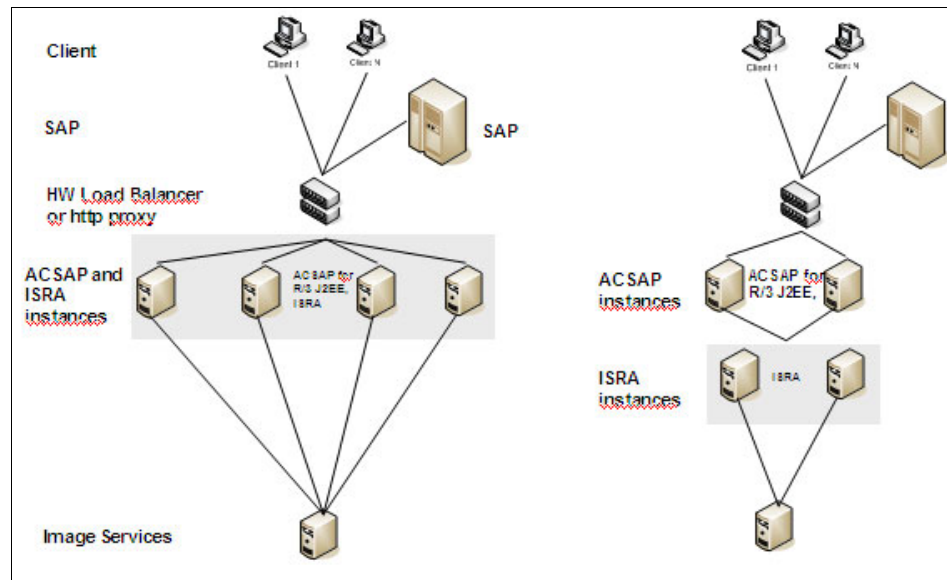


Figure 9-13 Scaling ACSAP for R/3 with Image Services

Similar to the Content Engine for the EJB transport, ISRA requires J2EE server-based load balancing. As a result, if a cluster of ISRA instances is needed to scale for handling the number of repository requests, you must use J2EE-based load balancing for the J2EE application server instances that run ISRA. However, because clients and SAP access ACSAP using the HTTP protocol, which is not balanced by the J2EE server-based load balancing, an additional hardware or software load balancer is required, as shown on the left side in Figure 9-13.

Alternatively, the J2EE application server cluster that hosts the ISRA deployments can be installed on separate servers or instances, so that the ACSAP instances are fronted by a hardware or software load balancer, and the ISRA instances are running in a separate J2EE cluster, as illustrated on the right side in Figure 9-13.

ACSAP for EP KM

ACSAP for EP KM is a solution for the SAP Enterprise Portal that is based on SAP's NetWeaver technology. It integrates the access to documents and their associated metadata stored in the ECM repository into the SAP portal.

The use case for ACSAP for EP KM is collaborative access to shared documents in the ECM repository from within the SAP enterprise portal. For this reason, the content elements are of the average size of a shared document, and there are commonly no background batch processes uploading large amounts of content.

ACSAP for EP KM consists of a Repository Manager for IBM FileNet P8 that is deployed on the SAP NetWeaver application server and connects the SAP Enterprise Portal Repository Framework to IBM FileNet Content Manager. The option, most frequently used to scale ACSAP EP KM, is vertical scaling by starting multiple instances of ACSAP EP KM on the SAP Portal server.

IBM FileNet Capture

IBM FileNet Capture enables capturing of paper-based documents and faxes and storing them as electronic images in the IBM FileNet Content Manager repository. Capture offers various options to provide metadata for the content, such as automatic extraction, using the ADR module or by manual keying using the indexing application.

Capture was designed to run as a distributed application and supports horizontal and vertical scaling for the different building blocks. Capturing images, performing quality checks, combining images into documents, and adding metadata to the documents before committing them to the repositories can include various steps. The sequence of this capturing process can be expressed as a *Capture Path*, which describes how the images flow through the different processing steps until they are committed as a part of the document. Capture does not require that a Capture Path is set up, but you can work with the different modules in an ad-hoc mode. To ensure that all steps are executed in the desired sequence, we recommend that you define a Capture Path for this purpose.

The approach to scaling Capture to process a larger amount of documents differs for the components that are used. Modules that require manual interaction by a person, such as the scan operation itself, the quality review, indexing, or indexing review steps, can be scaled horizontally by adding additional PC workstations to the installation. Images are scanned in parallel by the scan stations into separate batches that are managed in a database, and the Capture Path ensures that indexing and review stations pull images from these batches for further processing. By adding PC workstations, more people can work in parallel on the different batches.

Modules that perform automated tasks, especially ADR recognition components that extract data from the images using statistical methods can be scaled vertically, and adding processing power to an ADR server improves its performance and increases the throughput. Scaling horizontally by increasing the number of ADR servers is the other option that is available for these parts of the ADR installation.

IBM Content Collector

IBM Content Collector (ICC) consists of three major building blocks:

- ▶ An archiving engine
- ▶ Web applications and configuration service running on a embedded IBM WebSphere Application Server (eWAS)
- ▶ Legacy access components

ICC supports horizontal scaling for all components to increase the throughput for archiving new content and for serving a larger amount of users.

A horizontal form of ICC servers is called an *ICC cluster*. Because ICC runs on Windows operating systems only, there is no option to scale ICC vertically, except for using virtualization. ICC is a component that heavily uses the network to connect to the Content Engine for storing content. Therefore we do not recommend using virtualization to run multiple instances of ICC on a single physical server.

Figure 9-14 on page 287 illustrates the setup of an ICC cluster. The (primary) ICC server runs the core components. In addition, the configuration manager and the initial configuration template are hosted on this system. When additional servers are installed to form the cluster, they use the installation procedure for the expansion server, which only runs the core components.

All configuration for the ICC system is stored in a central configuration database that all servers in the ICC cluster access.

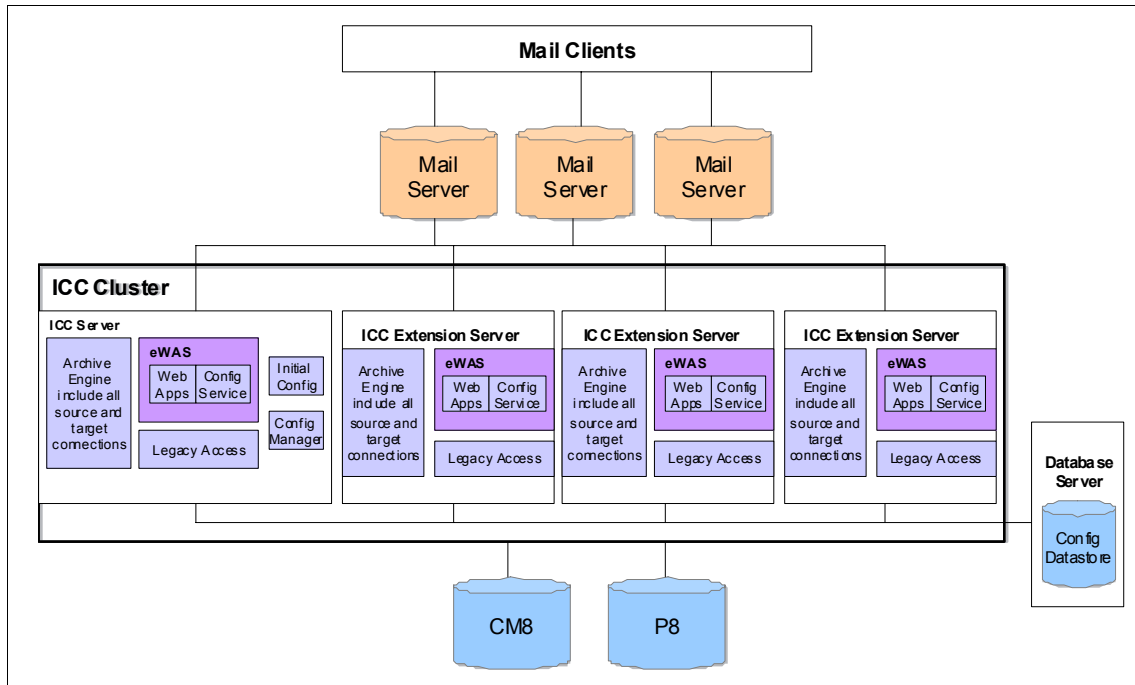


Figure 9-14 Horizontal scaling for IBM Content Collector

Scaling ICC: ICC performs retrieval by communicating directly with Content Engine; therefore, ICC must be scaled as utilization grows. This approach is different from its predecessor IBM FileNet Email Manager, which uses the Application Engine to retrieve and display content and thus relies on scaling the Application Engine to ensure speedy retrieval.

Connectors for Quickr and SharePoint

The connectors for Quickr and SharePoint consist of a repository connector component (the connector for SharePoint document libraries) that transfers content from the collaboration environment into the ECM system and extensions for the collaboration Web portal application, which allow direct access objects in the Content and Process Engine from the collaboration environment. These extensions are only used for retrieval purposes from the collaborative Web portal.

Collecting and transferring content

The connector for document libraries collects the content from the document libraries in the collaboration environment, transfers it into the Content Engine,

and (depending on the configuration) performs other tasks, such as creating a stub. The connector for Quickr document libraries currently allows you to collect content manually, not based on business rules, such as the SharePoint connector. For that reason, scaling mainly refers to the SharePoint connector. This component can be scaled vertically on a single server because the number of threads can be adjusted to the processing power of the underlying server hardware.

Additionally, if multiple Quickr or SharePoint instances are connected to the ECM repository, it is possible to distribute the overall load by installing and configuring individual instances of the connectors for a dedicated set of Quickr or SharePoint instances.

Retrieving content

The connector for SharePoint Web parts or Quickr Web portal forms the component that is used for retrieval purposes. From a technical point-of-view, it uses the CE and PE APIs to connect to the back end. Content retrievals are executed by calling the appropriate functions of the Application Engine UI Service.

The retrieval part can be scaled horizontally using a farm of SharePoint servers, for example. The Web parts are then installed on each server of the SharePoint farm, and therefore an increasing number of users and requests can be handled. Because the retrievals are triggered by persons working with the collaborative environment, increasing retrievals from the ECM system is typically seen if more users are working with the collaborative environment, so that a farmed environment might already be in place.

Because the content retrieval uses the Application Engine functionality, you must consider the impact of increasing retrievals on the Application Engine load. As illustrated earlier, the Application Engine can easily be scaled horizontally.

Process Analyzer

The Process Analyzer extracts data from the Process Engine event log tables and feeds them into its internal data warehouse. On a scheduled time interval, the data is aggregated into an internal data mart and from this representation OLAP cubes are derived, which can then be analyzed further by OLAP-aware tools, such as Cognos or Microsoft Excel. Converting data from the warehouse into the data mart and calculating the OLAP cubes is a labor-intensive job.

With IBM FileNet P8 release 4.0, the Process Analyzer can be scaled vertically to deliver higher throughput and faster calculation times.

The IBM FileNet P8 release 4.5 introduces partitioning for the Process Engine. Using this feature you can configure more than one Process Analyzer server,

which introduces horizontal scaling for the Process Analyzer. It can be defined, which Process Engine event logs are processed by which Process Analyzer server, for example, one Process Analyzer can be configured for each Isolated region. Using partitioning you can dedicate a Process Analyzer server for one or more event logs, across isolated regions, if required.

Process Simulator

The Process Simulator uses a process definition, arrival and work shift patterns, and probabilities for conditional routes to predict the flow of process instances, thus enabling it to perform what-if assessments to eliminate bottlenecks in existing processes or avoid them for planned processes.

The Process Simulator can be scaled vertically by adding additional CPU and RAM to the server on which this component is installed. Because the process simulator does not have to process items on the magnitude level of a production system, it is most likely that this component will not become a bottleneck.

IBM FileNet Business Activity Monitor

IBM FileNet Business Activity Monitor allows a real-time analysis of the Process Engine load. This feature allows management and executives to effectively monitor the KPI of business processes. It uses the Process Analyzer database and aggregates this data together with data from external sources in an in-memory database to derive and display the load and the KPIs.

The Business Activity Monitor server can be scaled vertically by adding CPU power and RAM.

9.2.7 IBM FileNet Image Services

IBM FileNet Image Services is an image management repository that was designed to efficiently handle very large amounts of electronic images. The Image Services architecture is optimized to deliver world-class storage and retrieval performance while managing billions of image documents. Content Engine uses Content Federation Services (CFS) to utilize Image Services as a repository for IBM FileNet P8. Figure 9-15 on page 290 illustrates the basic layers of the Image Services architecture.

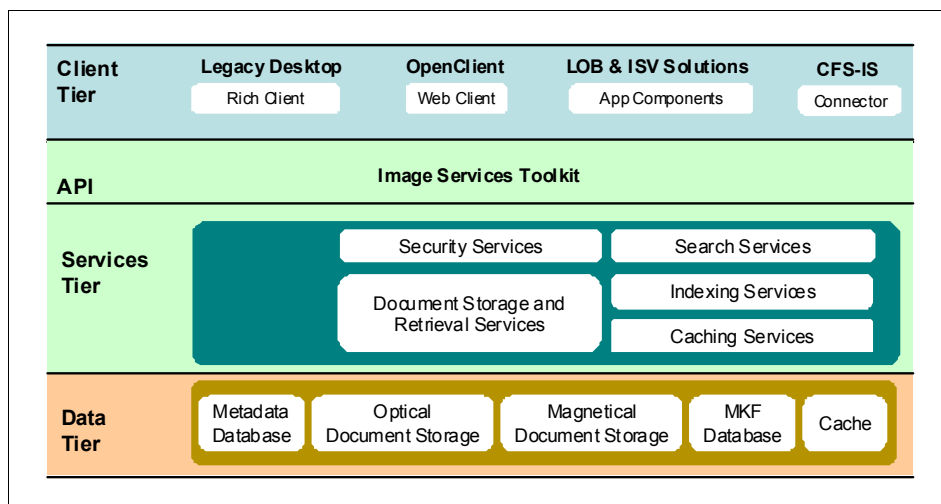


Figure 9-15 Image Services architecture

In Figure 9-15, the lower layer is the data tier that Image Services use to store the images that are managed and the data that is related to them. Some important building blocks are:

- ▶ The relational database that holds the metadata for the images.
- ▶ The multi key file (MKF) database that stores the location for each image on the storage media.
- ▶ The magnetic cache regions that provide very efficient batch document ingestion and speed up retrieval times, especially for documents that are stored on optical media.
- ▶ The optical and magnetic storage and retrieval systems.

Image Services supports a large variety of storage subsystem that can be used to store the images, such as Jukeboxes (optical storage), and several magnetic storage systems, such as disk, IBM DR-550 compliance storage, IBM N-Series with SnapLock, and EMC Centera. For a complete list of supported systems, refer to the *Image Services Hardware and Software Guide*:

ftp://ftp.software.ibm.com/software/data/cm/filenet/docs/isdoc/IS_HW_SW_guide.pdf

In Figure 9-15 on page 290, the services layer provides the services that are required to manage the stored documents. It consists of:

- ▶ Document storage and retrieval services that write and retrieve the objects to the storage system
- ▶ Indexing services that manage the associated metadata
- ▶ Cache services that manage the cache regions that are used
- ▶ Security services that provide access control to documents
- ▶ Search services that allow you to quickly locate documents based on their metadata.

Image Services are designed to work as a distributed application and are implemented as a set of logical services, which means that many of the components that are listed in Figure 9-15 on page 290 can be spread over different servers. In earlier times, when only optical media was used as long term storage, multiple OSAR servers were frequently used because they provided the required number of SCSI ports to connect a larger number of optical jukebox libraries. It is also possible to off load components, such as the indexing services or cache services, to one or more dedicated servers. Therefore, Image Services supports horizontal scaling.

Image Services is also optimized to effectively use the resources that the host server provides. As such, all Image Services component can be executed on a single system that provides vertical scaling. However, for systems that must connect to a large number of optical jukeboxes, the required number of SCSI controllers can be a limiting factor for pure vertical scaling.

9.3 Tuning the IBM FileNet P8 Platform for performance

For the IBM FileNet P8 infrastructure components that run in the context of a J2EE application server, at first glance, there is only a limited difference between the approach of scaling vertically versus horizontally because in both cases a farm of application server nodes is created. However, both alternatives show significant differences in that a hardware resource, such as memory or an I/O subsystem, must be shared between the different entities in the case of vertical scaling (comparable to virtualization).

There can be a benefit for vertical scaling because any communication between the components that run on the same physical machine do not have to travel over the wire. However, this does not apply if all Content Engine instances run on server1 and all Application Engine instances run on a different server2 because there will be no direct communication between the Content Engine nodes but

instead the majority of the traffic is seen either between Content Engine and the database and the storage subsystem and between the Content Engine and the Application Engine. Therefore, vertical scaling can show performance benefits if the IBM FileNet P8 core server components Application Engine, Content Engine, and Process Engine are running on the same physical machine, thus providing minimum latency and maximum bandwidth for the communication path. For installations that make no or only limited use of the Process Engine, this concept might be applied to the Application and Content Engine only.

Refer to the *IBM FileNet P8 4.0 Performance Tuning Guide* for additional information and recommendations:

ftp://ftp.software.ibm.com/software/data/cm/filenet/docs/p8doc/40x/p8_400_performance_tuning.pdf

9.3.1 J2EE Application Server

J2EE Application Server performance can be optimized in Java Virtual Machine and through connection pooling.

Java Virtual Machine (JVM)

The Content Engine and the Web application of the Application Engine are deployed into separate instances of a J2EE application server.

One critical aspect regarding performance is the garbage collection (GC). Garbage collection refers to an activity where the JVM re-organizes its memory heap, deletes all references to objects that are no longer used, and frees this memory for new objects. The larger the amount of memory that was configured for the JVM, the longer this garbage collection cycle takes and the more server resources are used for the garbage collection. The important point is that during the full garbage collection cycle of the JVM, all applications that run in the JVM do not respond. With 2 GB of memory, the garbage collection cycle can take 30 seconds or longer, depending on the resources that are available for the JVM.

On the other hand, the JVM needs a certain amount of memory to run the application. If there is a small amount of memory available, the JVM either often undergoes garbage collection cycles or throws an out of memory exception and eventually crash the application.

In general, the memory requirements of the Content Engine, and especially the Application Engine, strongly depend on the usage pattern. We recommend starting with a configuration of 1 GB of memory for the JVM of each Application Engine and Web application and 2 GB of memory for the JVM of each Content Engine.

To speed up the GC process, we recommend that you configure the generational concurrent GC strategy to reduce to the minimum the pauses that the GC cycles cause. On the downside, this strategy reduces the memory throughput to accomplish the improved GC cycles.

Note: For high ingestion scenarios, which create a large amount of short-living objects in the Content Engine, it can be beneficial to provide additional space in the tenured generations of the JVM heap by adjusting the ratios accordingly.

It is important to understand that different load patterns might require different JVM tuning to achieve optimal performance, for instance, it can be helpful to configure dedicated Content Engine instances for import purposes that run on J2EE application server instances that are optimized for high-ingestion performance; whereas, other Content Engine instances that are primarily used for retrieval use a different JVM configuration. If high volume ingestion is primarily performed over night, it can be beneficial to use scripts to shutdown the J2EE application server, change the JVM configuration accordingly, and then start the import jobs and switch back to an alternative JVM configuration for the daytime operations.

We highly recommend that you validate and monitor the effectiveness of the JVM performance tuning by using the appropriate tools, such as Tivoli Performance Monitoring (TPM) or similar.

Connection pools

The Content Engine uses the configured connection pools to communicate with the GCD and the object store databases. It is mandatory to adjust the `maximum connections` parameter for the data sources to the expected number of client connections. Refer to the IBM FileNet P8 Performance Tuning guide for detailed formulas to calculate the connection pool size dependant of the client connections expected.

9.3.2 Database

Content Engine and Process Engine use databases to store and retrieve information about content and process objects. It is important to configure the database accordingly to achieve optimum performance.

Database indexes

As a general guideline, it is important to know which queries are performed against the databases to create the required database indexes for preventing full table scans. Both Content Engine and Process Engine support the creation of

database indexes through the administrative tools Enterprise Manager and the Process Configuration Console. In many situations, simple database indexes are not sufficient because the queries that the API creates only benefits from complex indexes, such as a combined index. IBM FileNet P8 supports the creation of combined indexes for the Content Engine using the tools provided by the database vendor and for the Process Engine by using the Process Configuration Console.

Index skew

The distribution of values in an index might become uneven, for instance if half of the objects have the same value for an indexed metadata property. This situation is described as *index skew* and results in the database not using the index anymore and performing a full table scan instead, even for searches that would actually benefit from the index. By changing the query optimizer statistics strategy, as described in detail in the Performance Tuning Guide, the database can be instructed to use the index for those queries that refer to values in the index that are used only for a few objects.

Statistics collection

It is important to ensure that the statistics collection for the database query optimizer is run periodically and that the tools supplied by the database vendor are utilized, which helps to identify long running queries and suggest additional indexes to remedy these situations. In case the query optimizer is tuned manually, it is important to update the job profiles for the statistics collection on the database accordingly so that the changes are reflected and not overwritten with a default statistics job.

In high ingestion scenarios, using multiple object stores is beneficial because it allows you, at least, to address different tablespaces (DB2 or Oracle) or different databases (MSSQL) that can be physically located on different disks on the database server. Object store databases can also reside on different database instances or database servers that offer an additional degree of freedom for distributing the load.

9.3.3 Application design

Any performance optimization at the level of the IBM FileNet P8 Platform servers cannot remedy severe flaws in the application design or the data model of a custom implementation. The Performance Tuning Guide lists important recommendations to remember when designing individual applications. The Content Engine can effectively manage billions of objects. However, to ensure proper response times, the application must avoid critical queries or design patterns.

The Content Engine data model does not restrict the number of objects or subfolders that can be filed into a single folder. Retrieving all content for a given folder is a memory-intensive operation, if many objects are filed into this folder because the Content Engine stores the result set in memory to perform additional operations, such as access control. Therefore we recommend either limiting the number of objects (folders, documents, custom objects) that are filed into a single folder to a number of 200 maximum or ensuring, by application design, that retrievals to the content of folders that have to contain a much larger number of objects are properly filtered so that the retrieval does not return more than 100-200 hits.

Index skew: A single folder that contains a large number of objects (subfolders, documents, custom objects) is subject to cause index skew, as described in section 9.3.2, “Database” on page 293, because all objects have the same value for the indexed attribute `tail_id` in the containment relationship table.

The Content Engine API version 4.0 supports a paging parameter (continuable flag), which allows you to subsequently retrieve chunks of result sets. We recommend that you use this option carefully because it might negatively impact the performance. Let us assume a query that returns 10,000 objects. If this query is executed with the continuable flag set to true and a paging size of 50 objects, the database is still treated to retrieval all 10,000 objects and then sort the top 50 results for the first page. In such a situation, it is significantly faster to execute the query to the Content Engine using a `select top 50 ...` clause and turning the continuable flag off.

For performance reasons, we recommend cleaning up the Content Engine Audit Event table and the Process Engine Event Log tables on a regular basis. A best practice approach is to export all data that is related to work objects that were terminated and for content objects, which are linked to terminated workflows, if this data is needed for audit or compliance purposes and delete them from the corresponding tables afterwards. Again, the event-based architecture of the IBM FileNet P8 Platform helps to identify the correct objects for exporting and deleting.

9.4 Distributing an IBM FileNet P8 system

The modular nature of the IBM FileNet P8 Platform and its scalability are important benefits when designing ECM solutions. Using IBM FileNet P8 Platform you can install components of the complete system at different locations to address given infrastructural restrictions (networks) and to improve the overall

performance that the users experience. Throughout this section, we discuss how to approach the design of a distributed IBM FileNet P8 system.

9.4.1 Geographically dispersed users

Large organizations must support content management and content-centric processes over a large variety of geographical locations.

IBM FileNet P8 Platform offers a lot of configuration options that support the design of a distributed ECM system, and we briefly highlight the most important architectural building in this section. Refer to the *IBM FileNet P8 Distributed Deployment White Paper*² for a detailed treatment of the best practices for distributing an IBM FileNet P8 system.

Centralized IBM FileNet P8 system

In some cases a central IBM FileNet P8 system that clients access from different locations might be a valid solution. Alternatively, desktop virtualization technologies, such as Citrix or Windows Terminal Services, can also help to make applications available in various branch offices without installing servers locally.

Figure 9-16 on page 297 illustrates the architecture of a central IBM FileNet P8 system. All components are hosted at the central site, and the client at the remote site uses a wide area network (WAN) connection to communicate with the Application Engine.

The benefit of this configuration is the short distance between all of the core components, namely the Application Engine, Content Engine, Process Engine, and the storage layer (file systems for File Stores and databases). Additionally, managing a centralized system is easier compared to dealing with a distributed installation and its added complexity, especially backing up the data in a distributed topology can become a challenge.

² Available on request

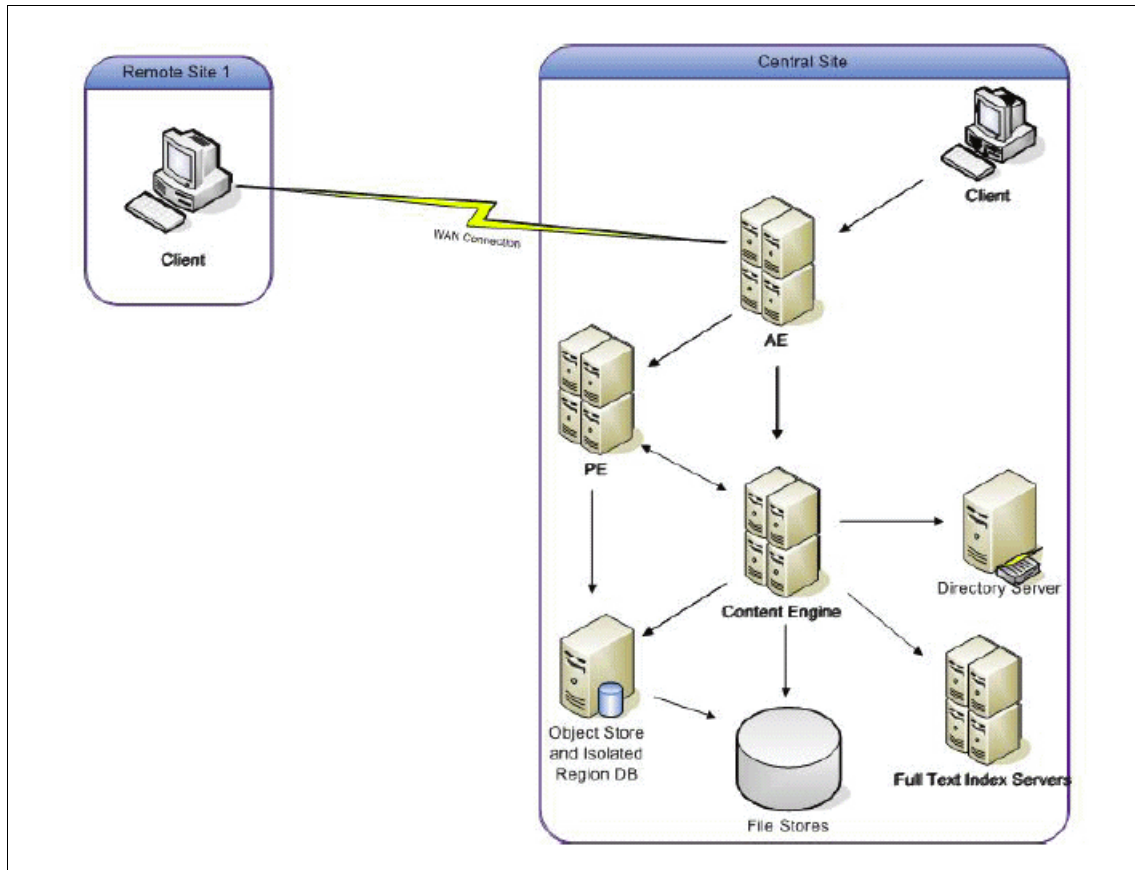


Figure 9-16 Central IBM FileNet P8 system

However, use cases remain that cannot be addressed by a centralized approach, for instance, in some countries legal rules exist that require that certain content must only be stored in the boundaries of the country itself. Another example is the work with large content objects, where a distributed storage is desirable to reduce the need to download the content from the central system to the remote location.

In general, we recommend starting with a central system and adding components at larger remote sites, if required.

Distributing Application Engine and Content Engine

The centralized approach can be easily extended by adding Application and Content Engine servers at a remote location, which is illustrated in Figure 9-17 on page 298.

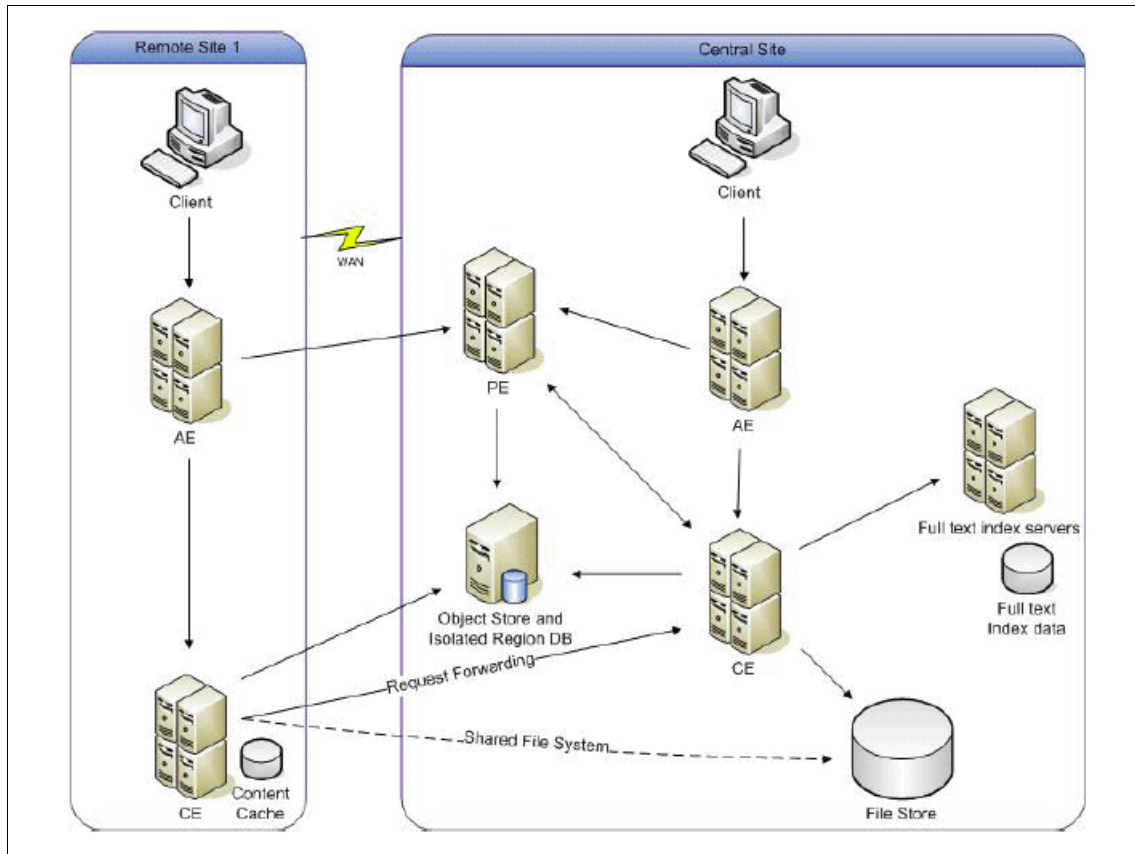


Figure 9-17 IBM FileNet P8 system with remote Application and Content Engine

In Figure 9-17, the client at the remote site communicates with the Application Engine at the same location. The Application Engine connects to the local Content Engine and the remote Process Engine. This configuration benefits from the Content Engine cache so that content objects that are created at the remote site automatically remain in the local cache.

Content objects can also be preloaded to the cache at the remote site if they are created at the central site. For this purpose, either the event-based architecture of the Content Engine is used or the prefetch can be initiated from a workflow by using an appropriate Java component. The preferred method depends on the question, when the decision can be made, and if the content object is processed at the remote site and a prefetch is required.

In addition, a feature called request forwarding can be exploited to improve the performance at the remote site. Request forwarding allows the Content Engine

that is at the remote location to forward queries to the central Content Engine for processing. The idea is that processing the request might include several round trips between the Content Engine and the object store database. By forwarding this request to the Content Engine that is located closely to the database server and only getting back the result set, there is less communication over the WAN connection.

Remote Application Engine only

It is also an option to only deploy the Application Engine at the remote site. In this configuration, the remote client benefits from a close proximity to the Application Engine, but the downside is the WAN connection between the Application Engine and the back end servers Content and Process Engine.

Because the Application Engine does not provide content caching, content objects might travel several times to the remote location, for instance, if they are requested more than once, which can be a problem, especially when working with large content objects.

Deploying only the Application Engine at the remote site has advantages, mainly for applications where the client and the Application Engine heavily communicate with each other and where latencies between the client and the Application Engine are significantly reducing the response times that the users experience.

Process Engine

Distributing the Process Engine is significantly more complex because of the nature of data that this component processes. Process Engine work objects are typically much smaller in size compared to the content objects, so that limited bandwidth between the remote location and the Process Engine Server does not impact the performance too much. However, the effect of latencies on the WAN is still encountered. Furthermore, for content-centric processes, the work objects are not only processed by human participants, but there are also interactions with external systems that, in most cases, also were not distributed.

Also, take into account that the Process Engine's high transaction nature results in a large number of communications between the Process Engine Server and the Process Engine database because all work objects that are handled are managed in the database. Thus, a distributed Process Engine requires a Process Engine database at that location, too. It is important to understand, that the caching concept for Content Engine also stores the content objects locally only, not the metadata information, which is because the information in a database is subject to change more frequently than the content itself. For the Process Engine, this distinction cannot occur because the process instances that are managed are typically small in size and underlie frequent updates.

We recommend using a central Process Engine and distributed Application and Content Engines for most scenarios, as shown in Figure 9-17 on page 298. This topology benefits from the number of communications between the Application Engine and the Content Engine towards the Process Engine. The Process Engine APIs are significantly less than the number of individual transactions that the Process Engine executes on the database.

Alternatively, if the process instances primarily remain in geographical regions, for example, when processing customer requests in North America and Europe, setting up different Process Engine systems for each region improves the performance, as illustrated in Figure 9-18 on page 301.

In Figure 9-18 on page 301, the master system is located in North America and consists of the Application Engine (AE NA), Content Engine (CE NA), and Process Engine (PE NA). A database server at this location (DB NA) hosts the Content Engine GCD and object store databases for North America and the Process Engine database for North America. A file server (FS NA) provides the storage areas for content in this location.

The other system is located in Europe and also consists of the Application Engine (AE EU), Content Engine (CE EU), and Process Engine (PE EU). Content is stored locally on a file server (FS EU), which either works as a pure content cache or also provides local file storage areas, depending on the detailed requirements. The process instances are managed in a local database (DB EU). European Object Stores are hosted on the database DB EU. However, the GCD information is always stored at one location only, in this case in North America. Access to content metadata for the remote location can benefit from request forwarding between the two Content Engines.

The downside of this approach is that work items cannot be transferred between both systems. Nevertheless, a client at one location can access the applications at both locations and can thus process the work items at the remote location (dotted line). However, when working with the remote Process Engine system, the access is over the WAN.

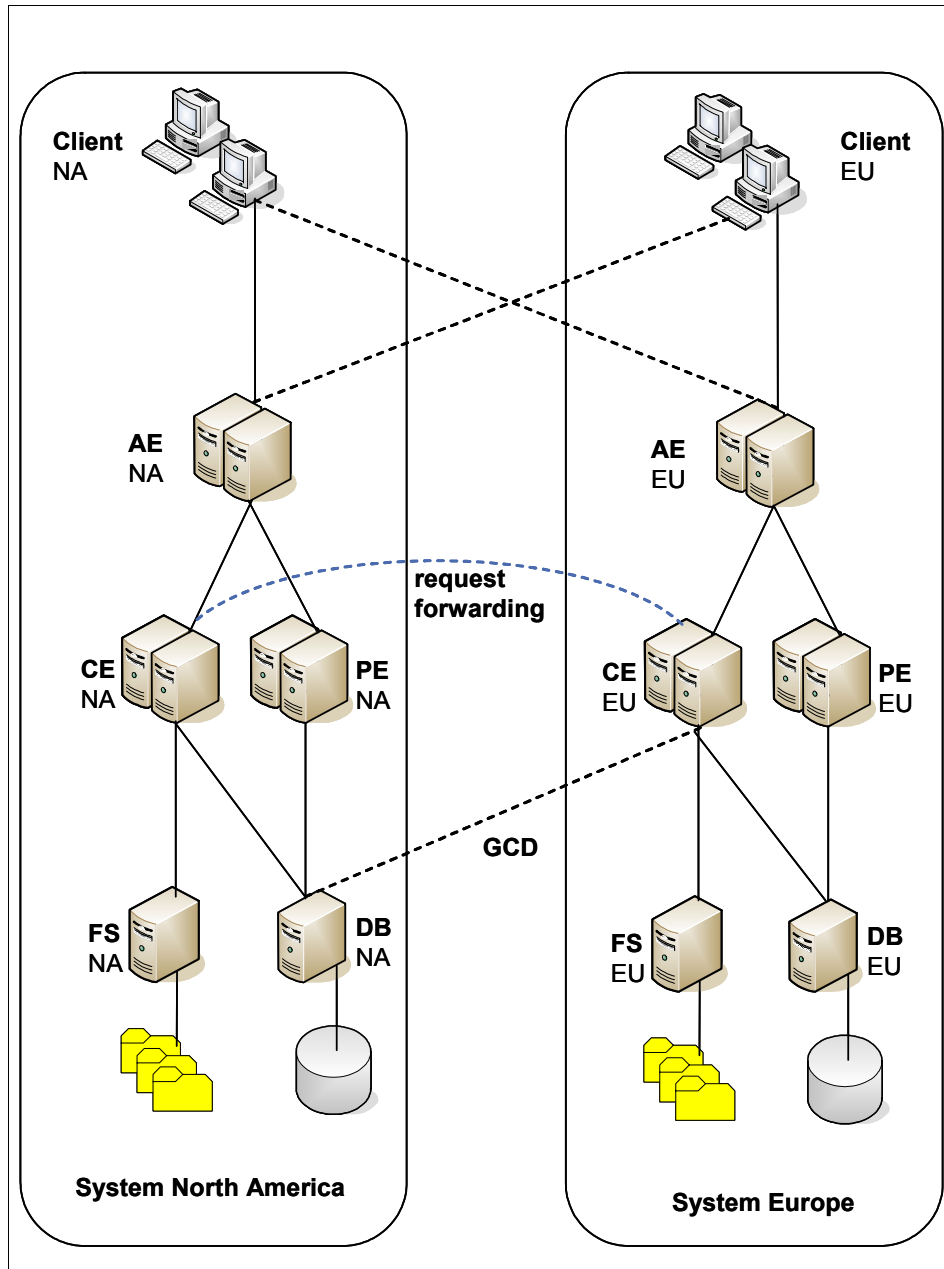


Figure 9-18 Distributed Process Engines

9.4.2 Disaster recovery

Disaster recovery (DR) addresses the loss of a complete data center or a complete site due to natural or man made disasters, such as fire, floods, or terrorist attacks. The goal of a disaster recovery strategy is to achieve business continuity with minimum interruption and data loss.

We do not want to go into the details of DR options for the IBM FileNet P8 Platform in this book because more details on this topic are discussed in the *IBM FileNet P8 Disaster Recovery Technical Note*³. We want to point out that disaster recovery and distributing IBM FileNet P8 systems are essentially two separate aspects. However, there are some interesting common aspects.

With the support of horizontal farming for all core components of the IBM FileNet P8 Platform, you might consider stretching the nodes server farms over a wider geographic area, for instance, in a metropolitan area network (MAN) with distances of up to 100 kilometers. Combined with mirroring the data layer (databases and file stores) such a configuration might theoretically address high availability and disaster recovery without the costs that are involved in implementing a dedicated disaster recovery infrastructure.

We do not recommend this simple approach because stretching the nodes of the farm introduces additional latencies for signals that travel over the wire and also introduce a new point-of-failure into the system topology, namely the MAN network. Typically this network is provided by an external service provider and is not as much inter control like the local area network (LAN), for instance, network lines might become interrupted by construction work, thus it will be required to implement redundancy by using multiple MAN connections, in the best case using different physical paths.

In addition, the MAN introduces additional latencies that are experienced in the communication between the different engines. We recommend that the distance between the two locations that are supposed to host the stretched farm be small (for example, 1-5 km) so that the impact of latencies is neglected, when considering the approach of a stretched farm. On the other side, such a close proximity might not be able to address full disaster recovery needs.

As described in the Technical Note³, the best practice approach for an IBM FileNet P8 system is to implement high availability using locally redundant components, such as a farm of servers, and to address disaster recovery by setting up a dedicated infrastructure at a secondary data center.

³ ftp://ftp.software.ibm.com/software/data/cm/filenet/docs/p8doc/40x/p8_400_disaster_recovery.pdf

Note: By combining the capabilities of modern load balancing devices and a sophisticated configuration, it is possible to reuse the disaster recovery hardware, for example, as a test or QA platform, and reconfigure it in case a disaster occurs.

9.5 IBM FileNet P8 in a DMZ environment

There are several use cases where an IBM FileNet P8 system is accessed not only internally by persons that are part of the organization, but also externally, for example by business partners or by customers, especially deployments that use business process management might incorporate access from external participants, for instance, a claim process might be started by a customer using a Web based self service portal. The customer is supposed to be able to track the progress of his or her cases based on a portal application. External experts can be part of the process, and they might upload content, such as pictures of the damage or reports into the case management system. Typically, organizations allow access for external participants to their internal systems only through a demilitarized zone (DMZ).

As described in Table 9-1 on page 250, a DMZ is typically an area that is secured and separated from the outside world (the Internet) and from the internal network by two different firewall systems. The aim is to reduce the risk that intruders from the Internet can get unauthorized access to the internal IT systems. To achieve the optimum protection, the protocols and ports that the firewalls route are restricted as much as possible.

Figure 9-19 on page 304 illustrates the general configuration for a DMZ with two firewalls.

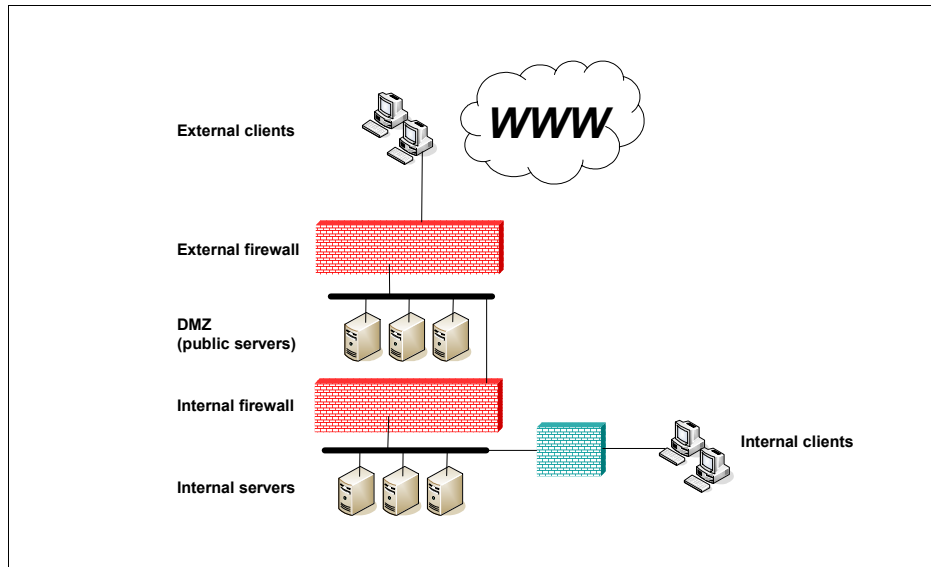


Figure 9-19 DMZ with dual firewalls

The internal clients can access the internal servers directly or through a separate firewall.

Web server with http proxy in the DMZ

The best practice approach for a DMZ deployment is to implement a clear separation between the IBM FileNet P8 servers in the internal network and the external clients outside of the DMZ by Web servers with http proxy capabilities. This reverse proxy must be transparent for both the external clients and for the IBM FileNet P8 servers.

In a typical scenario, the Web servers in the DMZ host some (in most cases) static external Web applications and also perform some sort of load balancing in the way that they distribute incoming request to the internal servers. These Web servers are typically not only used by the IBM FileNet P8 applications, but also by other applications that need access from external clients. The Web servers are configured to forward incoming requests based on the URL to the appropriate internal servers. Figure 9-20 on page 305 illustrates this scenario.

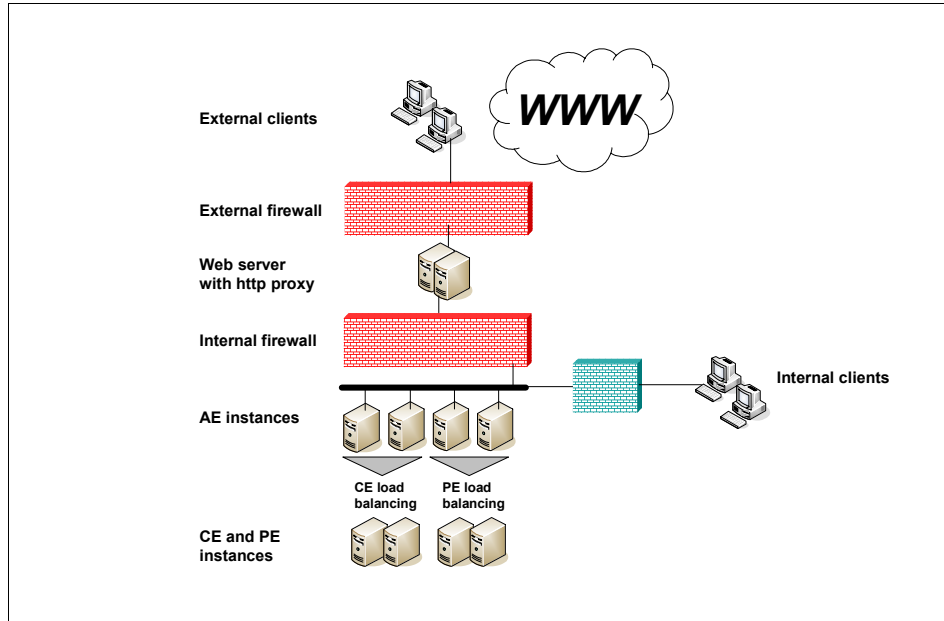


Figure 9-20 IBM FileNet P8 DMZ deployment best practice

For the external clients, the reverse proxy acts as the application server, and for the application server, the reverse proxy acts as the client. Therefore the reverse proxy must rewrite any packets that come from the external client in a way that they seem to originate from the reverse proxy server instead, which ensures that the application server passes the response to the reverse proxy server instead of trying to directly contact the client. Conversely, the reverse proxy must rewrite any response that it receives from the application server in a way that it seems to originate from the reverse proxy. This setup ensures that the client who evaluates the response reconnects to the reverse proxy for subsequent requests instead of directly trying to access the application server, which would fail because the application server is behind the internal firewall.

Application Engine in the DMZ

Under certain conditions, it is beneficial to put the Application Engine for external users into the DMZ. In this scenario, the Application Engine, or a farm of Application Engine servers, are installed at the level of the DMZ to serve the clients in the external network. Typically, there is another set of Application Engine Servers that run in the internal network that serve the internal clients and that also run the Component Manager instances. Figure 9-21 on page 306 illustrates this configuration. The other internal servers, such as Content Engine and Process Engine, are not shown.

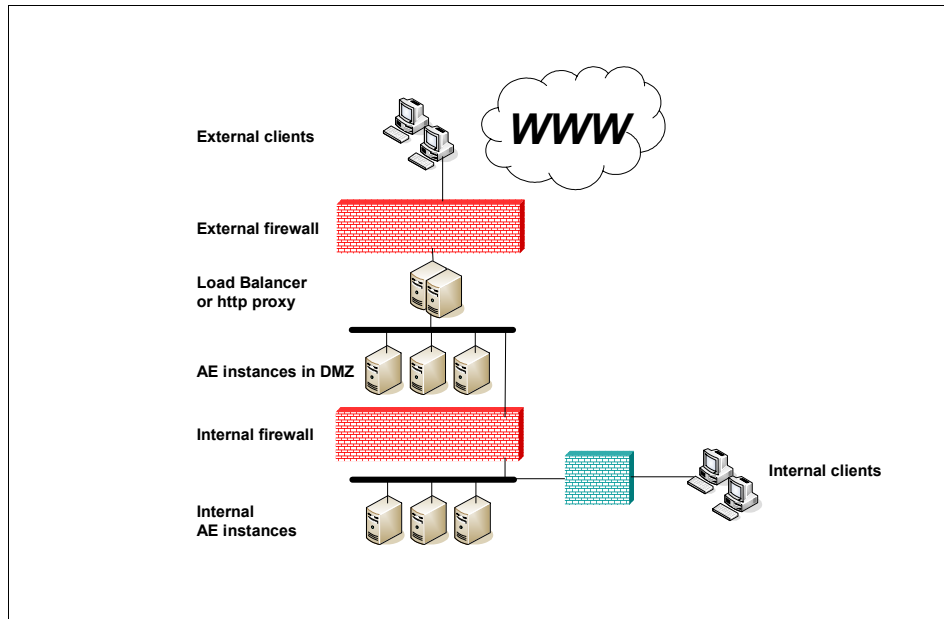


Figure 9-21 Application Engine in the DMZ

This architecture, in Figure 9-21, delivers the benefit that clients can directly access the Application Engine, which can be mandatory, for example, if HTTPS communication between the client and the Application Engine is intended to be used for external clients, and the HTTPS protocol cannot be routed across the DMZ.

9.6 Sample deployment

When designing an IBM FileNet P8 system, architects are not limited to exclusively adhering to only one of the concepts that we mentioned in 9.2.2, “Content Engine” on page 262, for scaling the Content Engine and managing the workload. Instead, in most cases we see a combination of different approaches to cover the full spectrum of applications. Typically, this involves application server work management for applications using the Java API and the EJB transport and hardware load balancing or an http plug-in for Web services clients.

In this section, for a sample deployment, we describe how virtualization can be used to effectively leverage hardware resources for a shared platform.

We assume that the following requirements are given for the sample deployment:

- ▶ Implementation of a shared infrastructure that serves multiple IBM FileNet P8 based projects.
- ▶ Template-style approach for using cloning to create the environment for a new application.
- ▶ Implementation of high availability (HA) for each application.
- ▶ Use AIX® as the operating system, and limit the number of LPARs.

Based on the requirements, the following system architecture is derived:

- ▶ For each application, a separate set of the three IBM FileNet P8 core engines are installed into one separate LPAR.
- ▶ Use another LPAR for each application that runs the database and the storage for this application.
- ▶ Use farming for the IBM FileNet P8 core engines (Application Engine, Content Engine, Process Engine) to ensure high availability.
- ▶ Use a traditional active passive HA cluster for the database and the NFS server (refer to the IBM FileNet Content Manager Implementation Best Practices IBM Redbooks publication⁴ for a detail description of HA clusters).
- ▶ Use two separate IBM p570 servers to achieve redundancy at hardware level and distribute the nodes of each farm across both servers.
- ▶ Use a set of two redundant hardware load balancers to distribute the load across the Application Engine and Process Engine farms and for the Content Engine WSI transport.
- ▶ Use application server-based work management for the Content Engine farms for the EJB transport.

Figure 9-22 on page 308 illustrates the basic idea of installing an IBM FileNet P8 domain into two LPARs. The LPAR named “LPAR application” hosts the IBM FileNet P8 core engines and is accompanied by a second LPAR (LPAR storage) on the same physical machine that provides the database and content storage.

⁴ *IBM FileNet Content Manager Implementation Best Practices and Recommendations*, SG24-7547

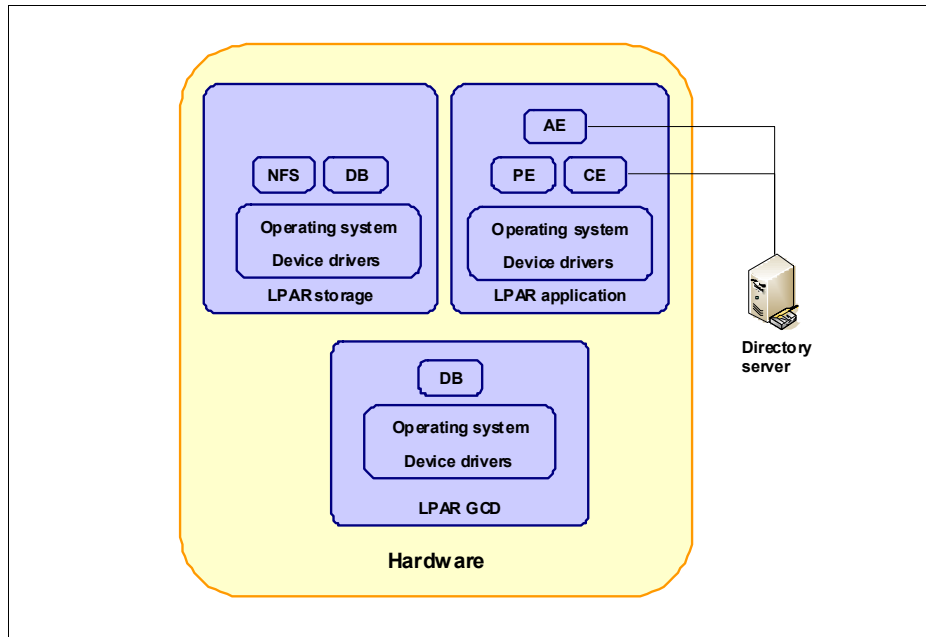


Figure 9-22 Design pattern for a clonable IBM FileNet P8 domain

The configuration in Figure 9-22 delivers the benefit of having building blocks that are fairly easy to clone to set up the environment for a new application. One central LPAR is used to host the GCD database for the domain.

By enhancing this approach with another LPAR application on the second server that hosts a second set of the IBM FileNet P8 core engines and configuring them as a farm for Application, Content Engine, and Process Engine, the requirement for high availability can be addressed on the application and engine level.

Introducing another LPAR storage on the second server and implementing the database and NFS server as HA clusters across both storage LPARs ensures that high availability is also considered at the level of the storage layer.

Figure 9-23 on page 309 illustrates these architectural considerations for an example of three different applications that run on the two servers (Server A and Server B) and are fully segregated from each other.

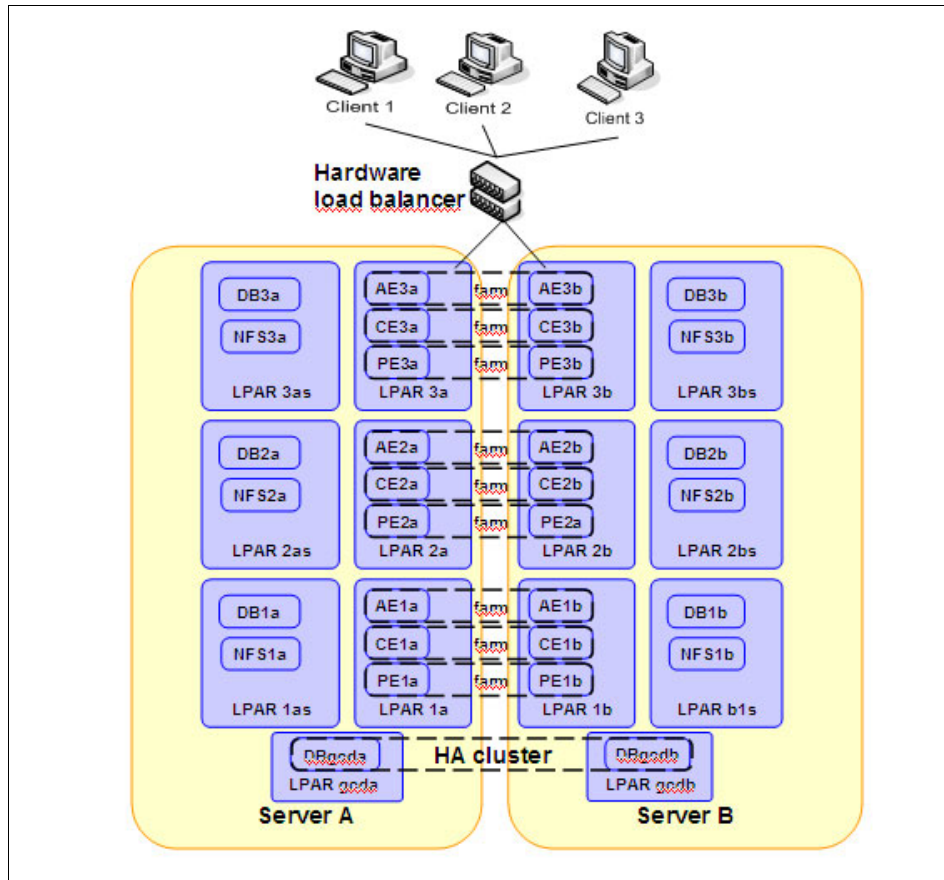


Figure 9-23 Virtualization used to implement high availability on a shared infrastructure

The application that is used by Client 1 uses the farms that are established by AE1a/AE1b, CE1a/CE1b and PE1a/PE1b. Client 2 uses AE2a/AE2b and so forth. In Figure 9-23, only the hardware load balancer is explicitly drawn. For the Content Engine Java API clients that run the EJB protocol, application server based load balancing is implemented across the Content Engine farms. Also, the HA clustering is only shown explicitly for the GCD database, even though it is also applied to any database that the applications use and for the servers that provide the NFS shares for the applications.

Refer to the solution templates that we describe in Chapter 10, “Architecting an IBM FileNet P8 solution” on page 311 for more information about typical IBM FileNet P8 deployments.



Architecting an IBM FileNet P8 solution

In this chapter, we use the features and advantages from the IBM FileNet P8 architecture that we discussed in the previous chapters and apply them to real-life customer solutions. We do this by approaching the problem as technical sales and enterprise content management consultants do by breaking down the business problem into key areas and addressing those areas with specific components and services of the IBM FileNet P8 architecture, and then architecting the overall solution to best fit the customer site.

We cover the following topics:

- ▶ 10.1, “Solution overview” on page 312
- ▶ 10.2, “Solution template: Customer Services Support” on page 313
- ▶ 10.3, “Solution template: Enterprise-wide Document Management” on page 319

Disclaimer: The scenarios that we describe in this chapter are fictitious. We provide them here for reference purposes only.

10.1 Solution overview

To help you architect an IBM FileNet P8 solution, we provide solution templates that show how some IBM FileNet P8 implementations are deployed, from a departmental system to a worldwide distributed system. IBM FileNet P8 implementation can grow over time, so the solutions that we present in this chapter are in the order such that growth over time is reflected because the architecture diagrams change between solutions.

We provide the following solution templates:

- ▶ Customer Services Support
Single site implementation. It has low volumes, has content, process, and case management. No high availability involved.
- ▶ Enterprise-wide Document Management
A worldwide distributed system for all documents in the customer facing and Human Resources departments. Heavy business process and content management use. Multiple offices distributed across the world's regions. This implementation uses scanning, electronic forms, and productivity suites that are integrated with the IBM FileNet P8 Platform to provide content creation.

Each solution template represents different areas to which the IBM FileNet P8 architecture can be applied, with differing sizes, user interfaces, and integration points, which allows us to discuss the specific points of business value and explain architectural decisions better in a practical context. The solution templates discussed are based on real life, live IBM FileNet P8 implementations.

For each solution template, we present the type of deployment, either small or large, based on our previous customer experiences. We add a mock business scenario around it to provide explanation as to why we choose to make certain solution and architectural decisions in the design. We also list the particular business problems to be solved and how features of the IBM FileNet P8 Platform solve the problems and provide business value over and above what the customer originally envisioned.

Each solution template consists of multiple sections. The solution overview section provides high-level information that is designed to address the main business problems, and identify products to be included and platform features used. As an extension of this, the future enhancements section goes beyond what is needed as part of the core solution to explore additional enhancement that can be made to drive additional value from a IBM FileNet P8 Platform implementation.

The bulk of the information is included in the solution architecture. The architecture section addresses customer and solution-specific issues that require the IBM FileNet P8 Platform to be architected in a particular way. These issues can be constraints, such as multiple sites, existing corporate software, IBM FileNet P8 products used, and sizing.

In addition to the purely IBM FileNet P8 architecture information, we also included, where applicable, information about existing industry process and data models. IBM FileNet P8 has direct mappings into the more content-centric processes and information stores of many of these models. This information provides a greater background to explain how an IBM FileNet P8 solution can fit into an organization's IT infrastructure.

10.2 Solution template: Customer Services Support

This is a single-site implementation without high-availability setup. It has low volumes of small documents (letters and eForms) arriving, each with a management process being launched. The solution requires content, process, and case management technology. Processes are relatively long lived to approximately 30 days.

10.2.1 Scenario

A water and gas utilities company is finding it hard to fix broken pipes and respond to customer requests. This difficulty is partially due to the number of customers requests, the inconsistent process being applied by different staff of different levels of experience, and inefficient paper-based processes.

10.2.2 Business problems and their solutions

Based on the business scenario, the company has the following business problems:

- ▶ Inconsistent processing
- ▶ Customer service level agreements not being met
- ▶ Hard to find documents and lost documents
- ▶ Storage costs increasing
- ▶ Costly customer servicing

Inconsistent processing

Inconsistent processes lead to missing information, and incorrect assumptions and decisions being made, which cost time to fix because technical staff must

second guess what the customer originally wanted, but do not have access to the original information.

Solution

Using IBM FileNet P8 *active content* technology, automatically initiate the correct complaint-handling process for a particular or general problem area, which removes initial manual handling and routing of the complaint.

Build the complaint handling process using IBM FileNet Business Process Manager. This makes the process well defined and removes the chance of it being carried out inconsistently.

Customer service level agreements not being met

The company is in a regulated industry and is subject to heavy fines if customer service requests are not handled in a timely manner.

Solution

When building the complaint handling process, configure timers to escalate work based on Customer Service Level Agreement (SLA) targets. Use IBM FileNet Business Process Framework to manage work items, and automatically prioritize work within an inbasket. Using this solution the company can merge cases where the same problem was reported by multiple people, which increase the information that we have about the reported problem and increase the speed in responding to it.

Hard to find documents and lost documents

Many customer requests are mislaid in a large area that is used for document storage. Also, files are often sent to people to process and are unavailable to others when needed or when on holiday.

Solution

Use Electronic Forms (eForms) to enable customers to report issues online. This solution ensures that the maximum useful information is recorded and does it instantly rather than waiting for paper to arrive. This electronic information ensures that the information is accessible to any permitted personnel at any time and that it is not being lost somewhere. It also means less paper to handle.

Storage costs increasing

The cost of storage increases due to more complaints.

Solution

Remove paper out of the company by introducing bulk scanning with IBM FileNet Capture ADR into an enterprise content management repository. Extract

easy-to-identify data, such as customer name, number, account number, date of report, and customer address, which makes instant savings of cost in searching for data and paper storage.

Costly customer servicing

Customers who phone in to ask about the status of their requests are hard to service, and the calls are often long because of the manual nature of the current work environment, which leads to high customer service costs.

Solution

Steps within the complaint handling process are configured to proactively inform customers of the status of their complaints, thus reducing the likelihood of them needing to call the customer services team and greatly improving efficiencies.

Using filters in IBM FileNet Business Process Framework, show active cases that match certain criteria. When customers phone in, it is easy to find their reports by searching by customer account number or area code. Also make customers' status available online such that they can look up the status on their own.

10.2.3 Customer architectural constraints

The company has about a hundred central staff located in a single office. There are currently 2000 letters coming in per day, Monday to Friday, arriving at 8:00 AM that need scanning as soon as possible for the workers to start handling them on the same day. This situation requires a high speed, bulk scanner and content repository that can handle high-peak ingestion rates and a constant rate of document retrieval during a day.

Integration with external systems requirement is to post code verification and e-mail systems and a SMS text message Web service.

The company's current operating environment is Windows with an Microsoft SQL Server database and Network Attached storage exposed as a CIFS share. The company is also keen to use existing IBM hardware saved over from a previous project. No disaster recovery or high availability is required.

The company has tight time lines and is keen to keep implementation costs low. The internal users will use browsers to process work. The external customers are also expected to use browsers to check their status. Therefore, no client-installed applications will be considered.

10.2.4 Solution architecture

Figure 10-1 shows the target single-site environment for this IBM FileNet P8 solution. Because of the relatively low demand in processing volumes and speed (per our sizing tool calculation, which we do not cover in this book), the Content Engine and Process Engine can be consolidated into one single server and still suitably under used at peak times.

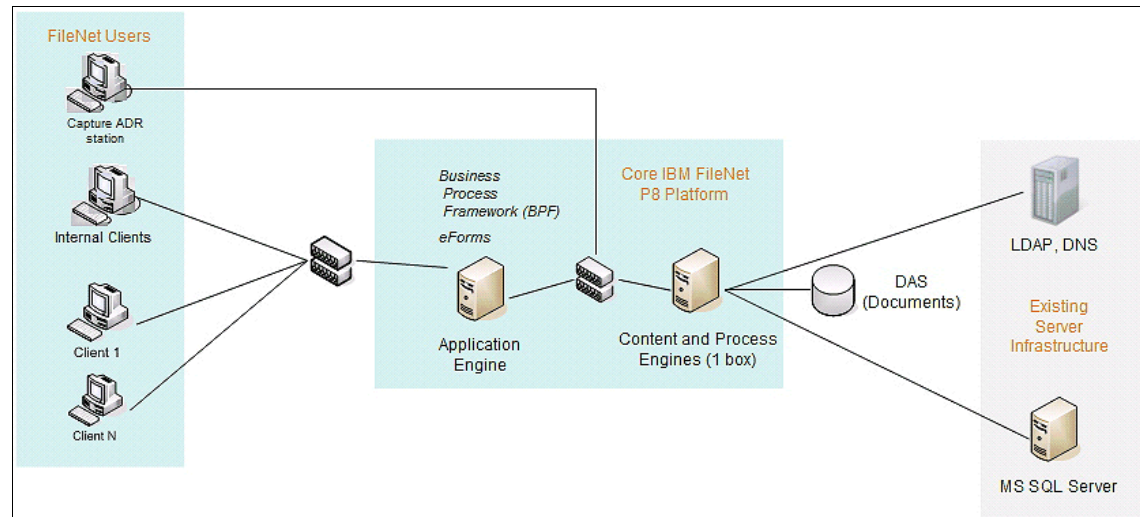


Figure 10-1 Core solution architecture

Target architecture assumptions

By adding eForms as a separate channel, we assume 40% of all future complaints enter the system through the customer or call center staff completing an eForms.

Systems hardware

The Content Engine, Process Engine, and Application Engine, MS SQL Server, and Directory Server are all on IBM System x® 306 P4 3.4GHz (Windows) - 1 CPU machines.

Note: IBM does not recommend that you co-locate all of the components because they each have different memory, CPU, and I/O usage profiles. This solution just shows that the power of a single machine is sufficient for this solution due to the scalability and performance of the IBM FileNet P8 Platform.

Sizing consideration

Although we do not cover the sizing of the system, we discuss some sizing-related consideration:

- ▶ Consider how long you will retain the content. We assume a content population of five years of operation to be retained.
- ▶ Be aware that the Content database grows large overtime because we want to record one case object for every document. In reality, cases can be merged where the correspondences for multiple complaints that report the same issue. So the overall case number is less.
- ▶ Consider whether to keep the case history audit fields for all time.
- ▶ Think about the information life cycle in any system to avoid creating a digital landfill of information that you never intend to use.
- ▶ Consider the additional processing demand when using eForms. By having eForms as an input mechanism, it increases load on the application server because the form is rendered once for filling in, and once later for review. This is different from incoming letters because scanning the letters includes a create document operation, rather than both a create document and render form operations as for eForms.

10.2.5 Solution processes

For this solution, we create four processes to handle the workload:

- ▶ Generic Customer Request Handling process
This is launched when we cannot automatically determine the type of complaint. The job goes to a unsigned work queue for someone to manually assign the type of complaint is used. The relevant process is then launched.
- ▶ Register Complaint process
When we know the customer request is a complaint, launch this process to extract complaint information and assign to the correct department. This automatically launches the Works Required process.
- ▶ Works Required process
This handles collation of similar reports and cases, assigning a works team, inspection, and correspondence with the reporting customers.
- ▶ Incoming Additional Correspondence process
The Works Required process might wait for additional customer correspondence. This process handles additional correspondence documentation, files it against the appropriate case, and re-awakens the main handling process.

As an example, Figure 10-2 shows the details of the Works Required process. In this process, we create a case with all of the relevant reports and process the case. For the processing case step, we assign a manager for the case, set the appropriate security, and service the client.

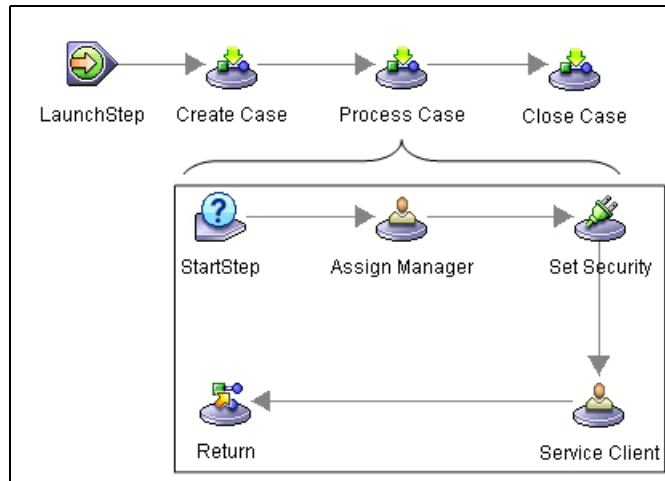


Figure 10-2 Works Required process

10.2.6 Future enhancements

For future improvements, the company can optionally do the following enhancements:

- ▶ Configure Process Simulator and Process Analyzer to identify process road blocks based on current and future staffing levels. Use this in the future to improve the process and make it more efficient. Also use the gathered information for business intelligence and reporting.
- ▶ Use Business Activity Monitor (BAM) to provide graphical feedback to managers of work at different stages and within or out of SLA targets. Also aggregate data by location to give a pictorial view of where problems are occurring.
- ▶ Use IBM Classification Module to determine the type of problem being reported, for example, a leak, incorrect bill, or loss of water to street. This makes deciding which team to send the customer complaint to be more automated, thus increasing work processing throughput.
- ▶ Integrate the business process with an existing customer information or CRM system. This solution gives process workers access to customer contact details other than those that are provided during reporting. This solution also enables the system to spot when details change so it can initiate a Customer

Details Validation and Modification process to make the internal data more accurate. Storing customer contact preferences means that your customer notifications can be sent through their preferred channel (phone, e-mail, SMS, or post).

If all of the above components were added to the final solution, then the core functionality is kept as it currently is, with the extra components shown in Figure 10-3 below added.

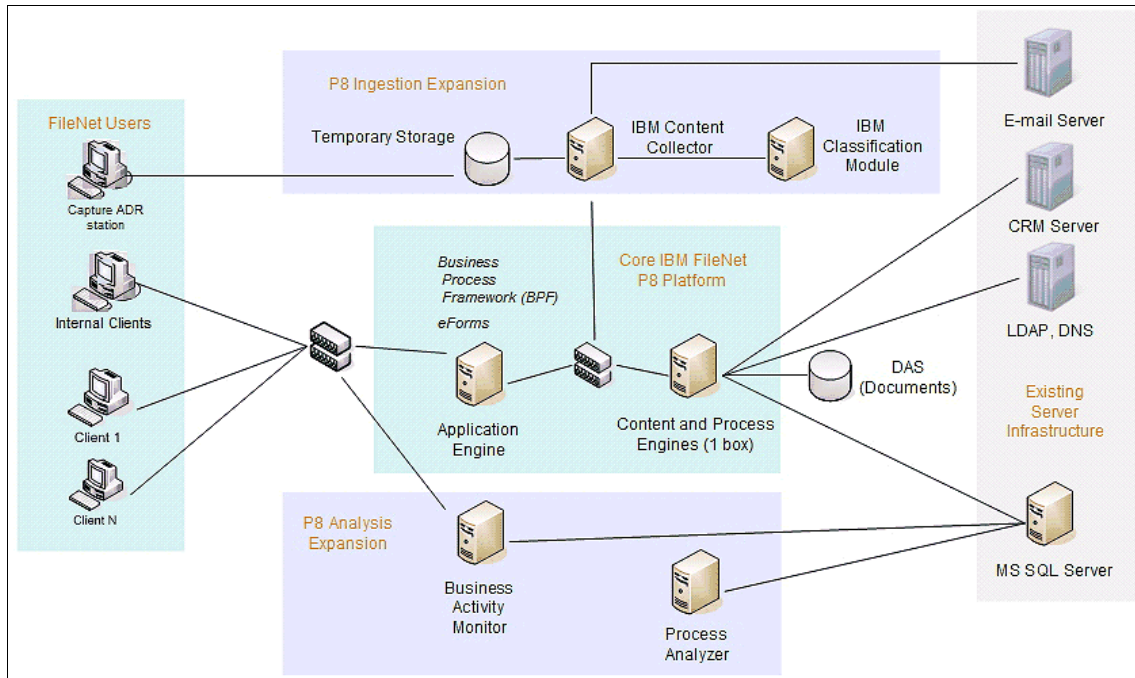


Figure 10-3 Solution with all optional components included

10.3 Solution template: Enterprise-wide Document Management

In this section, we discuss a worldwide distributed solution for all documents in the customer facing and Human Resources departments. The implementation involves heavy business process and content management use. There are multiple offices distributed across the world regions that use scanning, electronic forms, and productivity suites integrated with the IBM FileNet P8 Platform to provide content creation.

10.3.1 Scenario

The company is a multi-national company with offices and regional hubs around the world. They have a long history of using enterprise content management systems as departmental or country-wide solutions. They now want to move towards completely eliminating paper for all internal business functions.

This change involves capturing all customer-related documents, internal Human Resources information, policies and procedures, engineering documents, legal contracts, and all business-critical correspondence.

10.3.2 Business problems and their solutions

Based on the business scenario, the company has the following business problems:

- ▶ No single view of all customer data across all locations
- ▶ No consistent cost analysis or reporting on internal processes
- ▶ Inconsistent information security and retention policies
- ▶ Difficulty locating information
- ▶ Regional customization
- ▶ Reliability and scalability
- ▶ Development and support costs
- ▶ Uncontrolled file shares

No single view of all customer data across all locations

Information is fragmented and impossible to report. No single unified search for information across the organization exists.

Solution

We conducted an analysis of metadata and document classification required for customer related and Human Resources documents. The initial assessment found 60 document types with 200 metadata items shared between these classes. There are also 15 record types to consider with another 60 metadata items for these records management classes.

Future document classes will be subclassed from the above classes to enforce minimum required metadata standards. Search Templates and content indexing are set up to provide an infrastructure for finding information across all information stores and types.

These templates will also be used within business processes to find other relevant data based on initial information provided for each request, for example, a customer request for a new product would cause a business process to search

for the customer's last change of address and financial details documents. All of these are presented to internal users, which provides employees with instant access to all required information to complete their tasks without the need for manual search.

No consistent cost analysis or reporting on internal processes

Executives believe that the company can achieve a great deal of cost savings by analyzing current processes and automating tasks, where possible, which includes data validation, manual system updating, and paper-oriented tasks.

Solution

Perform a business process analysis for the Customer Onboarding, Account Opening, and Customer Maintenance processes. They identify steps that can be automated and made parallel and identify what information people must adequately perform human tasks.

Inconsistent information security and retention policies

There is a potential for confidential information to leak out from within the company. The company is keen to use security methods from preventing any unauthorized access. There is an existing hierarchical security classification method in the paper world that they want to apply to all content. Any leak of information could result in competitive, public opinion, financial, and criminal repercussions. These risks also occur when information is kept beyond the time it is required. Legal discovery costs hit the corporation hard in previous litigations.

Solution

The company's information security hierarchy is implemented within IBM FileNet Content Manager as a Marking Set and applied to the Security Classification property of all documents within the system. Each document class sets the default for this property to the most relevant setting.

Security Policies are created to act as application domain Access Control Lists. An example of this is a Human Resources Document whose policy allows all Human Resources users to read metadata, but senior Human Resources users to view the actual content.

Certain countries have legislation that prevent employee data being sent internationally. To comply, we create a marking set called Country Visibility and populate it when needed. This action effectively denies access to any out-of-country users, and is also very useful when dealing with security conscious government customers of the company.

Customer interactions generate cases within IBM FileNet Business Process Framework. All information is created to handle a customer request, and all correspondence are kept together in one case folder. After a vital business interaction, such as contract signing or account opening, is complete, these documents and the managing process are declared as critical business records to prove compliance.

Difficulty locating information

Employees find it hard to do their day-to-day activities because the information that they require is spread out between Web-based systems (including wikis and blogs), collaboration tools, e-mail, file shares, paper, and existing electronic repositories. The organization wants to replace these eventually, but first wants to make all of this content accessible to users through search and internal business processes.

Solution

Remove barriers in accessing information that is relevant to employees doing their job by providing federation at the content level to existing systems, migrating some systems that are not Web or ECM interface accessible (such as file shares), and linking to Web-based systems. Link this all together at the user interface level to provide all information that is required to make a decision on the same first summary window. Provide links to often used but not mandatory information, such as best practice guides, pricing rules, or Business Intelligence displays.

In addition to this process-orientated view of information, it is often necessary to search for answers across all existing repositories. Historically, search has been broken down across vast, and different sources of information because it cannot correctly categorize and index such diverse information domains. By taking the metadata and classification described in the point above, we can apply that to other information sources to make finding disparate but related data easier. This can greatly improve enterprise search result accuracy by providing information domain summary information to search by. At a simple level, this can be item types, such as news item or product information, but at a more detailed level you can aggregate information search results by, for example, country, product, or customer industry.

This solution also makes later transition to a common enterprise content management storage mechanism for all applications much easier to accomplish because all data is described using the same metadata schema.

Regional customization

Internationalization and localization are of paramount importance. The company promotes itself as an international player with local focus. As such, it tries to provide information in all languages, both national and local.

Information might be created in one world region, but primarily consumed in another. Bandwidth to some locations is limited.

Solution

By using the same underlying content and process metadata schemas, we can ensure that all content, regardless of origin, meets minimum indexing requirements. Many tools can be used to map country and language (locale) named data into this standard schema. An electronic form, for example, can have the same fields on it, but have a version translated from English with its left to right text into a right to left language with completely different local terminology and instructions. Other Web user interfaces, such as IBM FileNet Business Process Framework, can have language packs installed and detect the user's locale to show the interface in the most appropriate language.

Reliability and scalability

If a system goes down, it impacts directly to the company in terms of revenue generation and customer satisfaction. The company is keen to maintain the highest levels of uptime and resilience. They carry out over hundred millions of customer transactions per day. Every minute they are down, it costs them tens of thousands of dollars.

Solution

The content caching features of IBM FileNet Content Manager can be used to cache a document after the first authorized request at a remote location. So if a document in Los Angeles was requested by an employee working on the islands in the English Channel, for example, the first time they requested it there would be a lag. The next time any authorized user requested the same document, however, it would be drawn from the local cache.

An advanced feature of this cache is the write-through caching capabilities. This means if a user in New York checked in a document into the Los Angeles base object store, this document would be automatically cached in the New York content cache. This helps reduce the bandwidth usage and greatly reduce usage of the network for often used, but rarely changed, documents.

As we saw in previous solutions, we can provide horizontal scaling to meet high availability requirements. When a full site goes down, or is cut off from the network, a different approach is required. Disaster recovery can be provided by having a duplicate server infrastructure for a site, either active or passive, at

another location. This is made available to remote clients by making a network routing change and pointing all other sites at the disaster recovery site for this particular service.

Data integrity is maintained by providing background, behind the scenes, replication of file storage and database tables that the first site uses. This means that even if an entire site is unavailable for a full day due to network issues beyond the company's control, the alternative site would be made available within minutes, enabling the rest of the organization to work on this information and processes as normal, mitigating business risks due to down time.

Development and support costs

There are many existing systems that the company wants to eventually replace with off the shelf software wherever possible. As a stop gap method, however, they want to quickly replace the storage layer of their systems with an ECM persistence layer. They want to also ensure that in the future any new systems can use the same storage layer technology. Because the company is risk adverse, they want to ensure this layer completely abstracts the underlying content repository. This shields them from any future vendor specific API changes or changes of vendor.

Solution

Large organizations are increasingly looking at a Service Oriented Architecture approach to mitigate future proofing issues to do with software upgrades, dependencies, and migrations. As such, companies created a shared service business layer that abstracts content creation and retrieval to provide a managed shared service for their organization.

This results in a single independent interface using open standards to access content. It also makes the backend architecture transparent to the application. This is practical when you must monitor and charge for storage and access to other internal teams. It also makes migrating back end storage of content much easier to administer because you only need to update the mappings to storage on the shared service business tier, not in each and every application that might use the content, which you can achieve by using either a layer or an enterprise content management system that supports SOA and federation, such as the IBM FileNet Content Manager. You can use the IBM FileNet P8 stack with its built in Content Federation Services (CFS) mappings to other content management and heritage systems. The additional advantage, other than the maintenance cost advantage, of using CFS is that the add-on products all then have visibility over the content. This visibility is particularly useful for adding process accessible content to your application that is built on top of IBM FileNet Business Process Manager and IBM FileNet Business Process Framework. In this scenario,

federation, as opposed to a intermediary service API, can achieve greater performance and usage features.

IBM FileNet Content Manager and IBM FileNet Business Process Manager are fully accessible, which we learned in this book, through Web services and other APIs. You can even invoke and interact with individual running processes using Web services. This fits perfectly into a service-oriented architecture, as required by the Shared Service or Software as a Service (SaaS) models.

Uncontrolled file shares

There will be a migration from existing departmental and user file shares into the new system to remove duplicates and reduce storage capacity. There are currently billions of documents over 50 file servers. Bulk classification is error prone, whereas manual classification is too costly to perform. The company cannot find a good solution to this.

Solution

IBM Content Collector for File System is used to migrate the more well defined departmental file shares. Rules are developed to classify documents of particular types, names, and filing locations into specific classes. The unstructured user shares, however, require a more flexible and automated approach. IBM Classification Module (ICM) can be used with IBM Content Collector to suggest the most likely class and filing location for the documents on the share.

As we mentioned in 6.2, “IBM Classification Module” on page 130, IBM Classification Module works by teaching its neural networks with a corpus of documents of a specific classification and filing location. The larger the corpus you train with, the higher the accuracy. Any documents that the system is unsure about will be routed for manual processing. In this situation, the user sees the suggested classes and filing locations and either agree to them or choose an alternative. As IBM Classification Module learns to be more accurate over time, the accuracy score will improve, causing less and less documents to pass through the manual process.

You might choose, for example, to get 100 internal employees to ingest 100 of their documents and provide the correct classification and filing locations. This results in a training corpus of 10000 documents. You can then use this as a basis for the automated migration, perhaps migrating a proportion of your users’ content per week.

After the migration is complete, the ICM system can continue to be used to match new incoming documents. At this stage, it is very well trained, and can be used to help users classify new documents, or by the system for incoming emails or OCR text from scanned images. This greatly reduce user error, increase employee buy-in for using the system, and prevent users from sticking to what

they are used to, that is continuing to use personal storage and file shares for new documents.

10.3.3 Customer architectural constraints

Preferred large scale system is IBM p570 with AIX and LPARs to support virtualization. Other preferred internal systems include Tivoli Directory Server, Tivoli Access Manager with WebSeal for SSO, WebSphere 6.1 Application Server, DB2 RDBMS.

Major regional centers include Los Angeles, London, Dubai, Hong Kong, Johannesburg. Offshore customers handled out of New York, Channel Islands (English Channel, UK). This connects to the London regional hub. 50 smaller offices throughout each region. Los Angeles, London, and Hong Kong, each handle approximately 100000 employees, with Johannesburg and Dubai taking approximately 50000 each in their regions.

The company has 20 million customers worldwide ranging from individuals up to large multi-nationals. On average 10 million customers sign up for new products and services every year, with each request requiring an average of 10 documents of 40KB being supplied by the customer for the onboarding process. 80% of these documents are eForms. A further 20 documents are generated internally (10 eForms approvals, 10 documents to be sent out) on average for each request before completion. Individuals account for 95% (9.5 million) of the company's new requests, with the remainder (5%, 0.5 million) being from companies.

The company has 250000 employees. Internally the company has a team of 1000 Human Resources professionals (250 senior employees). Focus has 20000 people managers, and 100000 employees working on customer facing duties.

The client onboarding process consists of 10 independent business processes with an average of 15 human steps. Of these, 5 are automated in the to be process; 2 are data validation, 2 system queries, and 1 is a system update. There will be an additional 10 system steps to manage content. It currently takes 30 days to complete a product onboarding request.

There are Human Resources Onboarding, Employee Leaves, Change of Details, Change of Manager, Change of Role, Employee Review, Disciplinary Action and Document Access Request processes. Typically there is a 10% attrition rate among employees. Onboarding requires 10 documents be imported or generated (5 of these eForms), Employee Review and Disciplinary Action 5 (all eForms), the rest 1 (all eForms). On average, these processes have 5 human steps. Of this, 2 are information validation and discovery steps.

The document access process is currently manual and paper based and is used 600000 times a year. It also takes two weeks to complete. These both need reducing by setting default document security and implementing new streamlined employee leaving, change of role, and change of manager processes. The target processing time is five days.

10.3.4 Solution architecture

Disaster recovery can be achieved by mirroring each site at a nearby, but independent facility. Set up disaster recovery sites as hot standby and normally do not receive user requests. High availability are designed into each site's infrastructure.

We call Los Angeles, London and Hong Kong high load sites, and Dubai and Johannesburg medium load sites.

For load modelling, we assume that all client onboarding customers are spread throughout each region, according to the number of employees at each regional office (the regional staffing is directly proportional to the number of customers in that region). These requests are handled on the new follow-the-sun operating model.

All Human Resources requests are handled in region and do not follow the follow-the-sun model, which are spread out as per the number of employees per region.

Figure 10-4 on page 328 shows the large site system architecture. We use mainly IBM P570 machines. The number of instances and assigned CPUs are based on sizing from Scout (which we do not cover in this book). We also have not included IBM Classification Module (ICM), IBM Content Collector or Content Federation Services (CFS) in the solution diagram. In our scenario, we only talk about using these for migrating content.

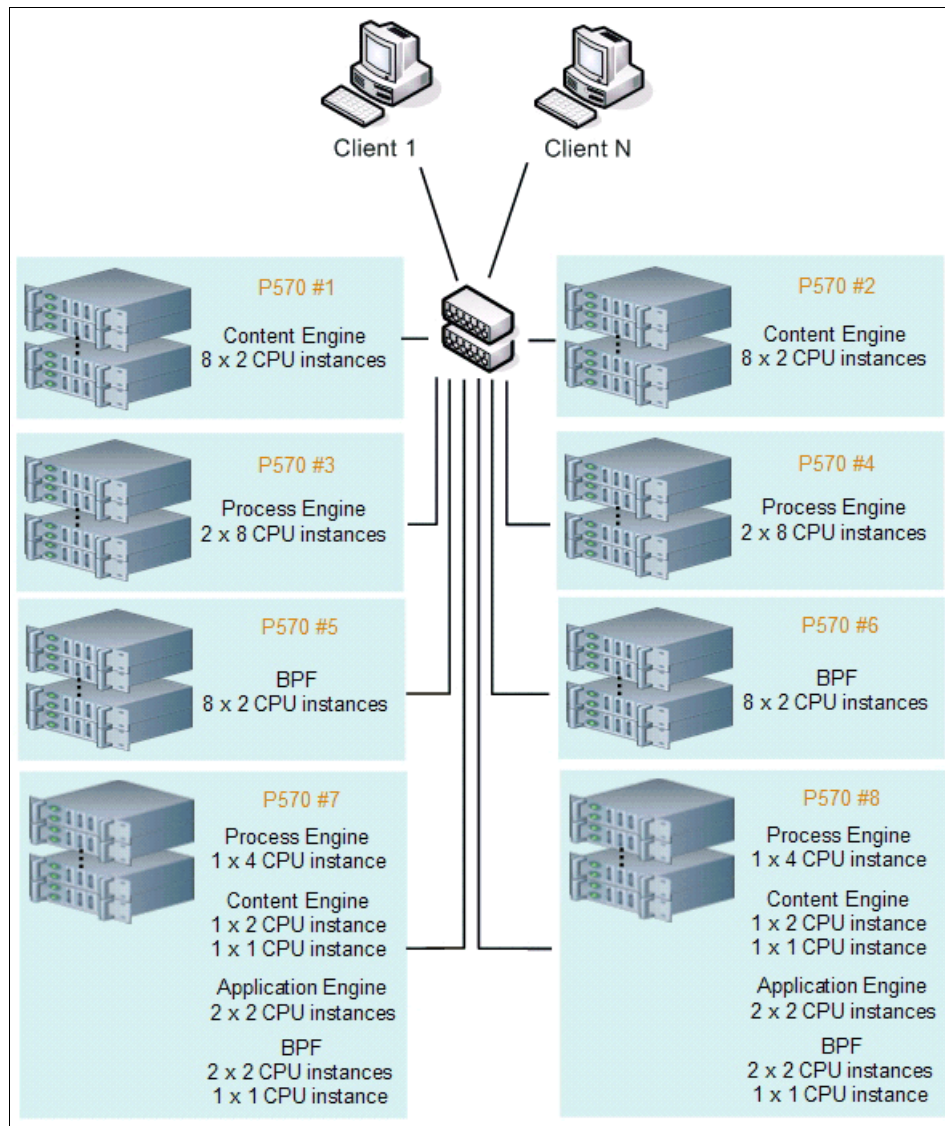


Figure 10-4 Large site system architecture

In Figure 10-4, we only show servers from a sunny day scenario, which means that we assume that no servers ever fail and thus we have not installed a highly available service. In practice, thanks to clustering technologies, making a service highly available is easily facilitated by adding extra load handling nodes and have an automatic failover mechanism that is transparent to the client. Having a

network layer with virtual hosts for the clustering and farming mechanism makes this transparent to the client application.

Related publications

The publications that we list in this section are considered particularly suitable for a more detailed discussion of the topics that we cover in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 332. Note that some 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
- ▶ *Understanding IBM FileNet Records Manager*, SG24-7623
- ▶ *IBM High Availability Solution for IBM FileNet P8*, SG24-7700

Online resources

These Web sites are also relevant as further information sources:

- ▶ IBM FileNet P8 Platform main 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>
The above URL includes links to all expansion IBM FileNet P8 products.
- ▶ IBM FileNet Content Manager
<http://www.ibm.com/software/data/content-management/filenet-content-manager>

- ▶ IBM FileNet Business Process Manager
<http://www.ibm.com/software/data/content-management/filenet-business-process-manager>
- ▶ IBM FileNet Records Manager
<http://www-01.ibm.com/software/data/content-management/filenet-records-manager/>

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

A

- access control 141, 147, 187–189, 193, 291, 295
 - help 148
 - information 147–148, 156
 - permissions 196
 - specification 209
 - support 189
- Access Control Entry 32, 193, 196, 205
- access control list 189, 193, 196
- access control list (ACL) 32
- access roles 195
- account manager 219–220, 240, 242, 244–245
- ACSAP 282
- ACSAP for EP KM 284
- ACSAP for R/3 282
 - scaling 283
- action event
 - asynchronous 160
- active content 9, 56, 67, 142, 146, 151, 154, 314
- Active Directory 105
- ad-hoc correspondence 247
- ADR 76
- advanced capture process 69
- Advanced Document Recognition (ADR) 58, 68
- advanced search 104
- agile enterprise content management 139
- Analytics Engine 46
- APIs 113, 137, 145, 147, 325
- Application Connector 12
- application connector 82–83
- Application Connector for SAP R/3 281
- application design
 - performance 294
- Application Engine
 - distributed 297
 - DMZ 305
 - scaling 257
 - scaling options 23
 - WAN connection 299
 - web application 282, 292
- Application Engine (AE) 7, 17, 28–29, 47, 113, 187, 231, 257–258, 281, 316
 - system architecture 48

- application framework 128
- Application Frameworks products 57
- Application Programming Interface (API) 6, 250, 261
- application server 28–29, 86, 134, 188, 190, 230, 251, 256, 282, 317, 326
 - Content Engine APIs 263
 - ERP interfaces 86
 - multiple instances 263, 282
 - servlet container 257
 - web container 257, 262
- architecture
 - Application Engine 48
 - BAM 126–127
 - BPF 123
 - IBM Content Collector 62
 - Process Engine 40
- Archive Engine
 - architecture 63
- asynchronous action event 160
- Asynchronous JavaScript and XML (AJAX) 113, 123
- attachment 166
- audit 120, 157–158
- audit log entries 157
- auditing 149, 157, 175, 233
- authentication
 - Content Engine 32
- authority 147
- authorization 202–203
 - Content Engine 32

B

- BAM
 - architecture 126–127
- bar code 73
- basic search 104
- blank page detection 73
- BPF
 - architecture 123
- BPF Explorer 117, 123
- BPF Layout Designer 118
- BPM 166–167

- BPM system 123, 171
 - common requirements 173
 - complete process 171
- building block 153
 - horizontal and vertical scaling 285
- bulk scanning 314
- business activity 3, 10, 108, 124, 175
- Business Activity Monitor 46, 108, 125, 128, 176, 289, 318
 - server 289
- Business Activity Monitor (BAM) 57, 175–176
- Business Intelligence (BI) 15, 318, 322
- business process 2–3, 40, 56, 68, 75, 82–84, 108–109, 121, 141, 146, 151, 154–155, 168, 211, 289, 312, 314, 320
 - active, integral part 82
 - optimization 180
 - optimize 128
 - status 176
 - update 223
- Business Process Framework 114, 118–119, 209, 220, 223, 239, 247
 - scaling 280
- Business Process Framework (BPF) 57
- business process management (BPM) 160, 166
- Business Process Management Suite (BPMS) 194
- Business Process Manager 28, 112, 124, 134, 166
- Business Rules Engine (BRE) 174

C

- Capture 58, 67
 - scaling 285
- capture
 - advanced capture process 69
- Capture ADR 76–77, 161
- Capture Advanced Document Recognition (ADR) 75
- Capture Desktop 71
- Capture integration 78
- Capture module 75
- Capture operations to manipulate (COM) 73
- Capture Path 68, 72, 76, 285
- Capture Professional 67, 70
- Capture Toolkit 72
- case management interface 120
- case object 122, 124, 239
- catalog 145
- category 131

- CE Java API 267
- CE_Operations 223
- CEWS 262
- CFS-IS 22
- challenge
 - content 2–3
- classification 38, 60, 132, 161–162
- Classification Workbench 133
- client tier 39, 46
- Cognos Now 57
- Common Base Events (CBE) 177–178
- Common Internet File System (CIFS) 35
- communication
 - encrypt 230
- compliance 139
- Component Integrator 18
- Component Manager 42, 50, 160, 170, 257–258, 260–261
 - large polling interval 262
 - multiple instances 261
 - separate instance 261
 - single instance 260–261
- Component Queue 195
- component queue 42
- Component Relationship Objects 153
- compound document 152
- Compound Document Framework 153
- connector 82–84, 100
- Connectors 13, 56, 63
- connectors 100
- connectors and federation products 56
- consumer
 - web services 44
- content
 - challenge 2–3
 - classified 106
 - controlled 106
 - federated 146
 - ingestion and classification 132
 - reused 106
 - view permission 193, 203–204
- content based retrieval (CBR) 272
- content element 32, 34, 145–146, 150, 269–270
- Content Engine 7, 28–29, 187, 193, 250–251
 - action handler 241
 - administration client 39
 - administration tool 39
 - API 223, 267–268
 - application 19, 30, 40

- audit event table 295
- auditing 149
- cache 298
- caching concept 299
- case 175
- catalogue 146
- CBR Executor 273
- class 215
- client 250
- connection URI 238
- connection URL 238
- content federation layer 146
- custom objects 239
- data model 295
- deployment 39
- deployment process 266
- diagnostic information 234
- distributed 297
- Document Class 161
- EJB transport option 231
- EJBs 265
- event 159–160
- event action 158–159
- event framework 33
- Event object 234
- farm 265, 267–268
- federates content 22
- fixed content devices 35
- folder 51
- folder instance 151
- functionality 173
- GCD 300
- high availability 266
- IndexArea object 37
- information 146
- instance 261, 263, 265, 271
- integral part 263
- integration 171
- Java API 231, 265, 283
- Java API client 309
- Java event handler check 246
- layer 269
- lifecycle actions 164
- load balancing 265–266
- load distribution 269
- load management 265
- logging 234
- logging configuration 235
- metadata model 185
- node 262, 266, 278, 291
- object 31, 33, 193, 198
- object level JDBC provider 238
- object store 40
- object store database 217
- performance 271
- property 218
- repository 18, 49
- request forwarding 300
- scaling options 23
- search templates 185
- security 146
- security model 194
- security templates 211
- server 30, 33, 159, 250, 263
- server instance 30
- server resource 264
- server-side behavior 34
- single deployment 265
- stateless, farmable architecture 36
- storage capabilities 50
- storage service 36
- store 295
- subscription processor 159
- system architecture 30
- system events 156
- Web Service 269
- Web services API 192
- Web services APIs 231
- WSI transport 307
- XML files 150
- Content Engine (CE) 17, 28–29, 32, 67, 74–75, 78, 86–87, 104, 111, 141, 145–146, 148, 155, 170, 192–193, 196, 250, 259, 316
 - audit 157–158
 - authentication 32
 - authorization 32
 - internal database structure 31
 - object store 104
 - security 105
 - storage options 35
 - storage services 36
- Content Engine event
 - action 164
 - component 159
- Content Federation
 - Services for Image Services (CFS-IS) 22
 - strategy 20
- content federation 20, 145

- content federation layer 146
- Content Federation Service 145, 184, 263, 289, 324, 327
- Content Federation Service (CFS) 20
- content ingestion products 56
- Content Management
 - Interoperability Service 91
- content management 1, 55–56, 82, 87–88, 141–142, 145, 166, 186, 275, 296, 324
 - multiple point solutions 19
 - store and retrieve 143
- Content Management Interoperability Services 56
- Content Manager 28, 50, 112, 136
- content object 34, 144, 146, 270, 272, 278
- content repository 101
- Content Search Engine 36, 272
 - scaling 272
 - system architecture 37
- content storage 34, 164–165, 270, 307
- content-centric process 167
- core engine 17, 103, 143, 192, 230, 257, 276
- custom application 16, 19, 46, 48, 158, 162, 175, 202, 209, 211, 222
- custom event 158
 - action 34, 159–160
 - component 159, 183
- custom event action 159
- Custom Object 31, 33, 151, 193, 198, 200, 202, 215, 295

D

- Darwin Information Typing Architecture (DITA) 153
- Dashboard 125
- data model 31
- data type 166
- database
 - Process Engine 41
- database farm 218
- database indexes
 - performance 293
- database schema
 - Process Engine 43
- database storage area 270
- database structure
 - Content Engine 31
- decision point 59
- Demilitarized Zone (DMZ) 229
- Department of Defense (DoD) 50, 52

- dictionary 131
- disaster recovery (DR) 255, 302, 315, 323
- discovery 139
- discovery and compliance products 57
- disposition 225
- Disposition Sweep 225
- distributed system 296
- distributing
 - Application Engine and Content Engine 297
- DMZ 229–230, 250, 260, 303
 - Application Engine 305
 - deployment best practices, P8 system 305
- document access 226
- document assembly 74
- document class 34, 72, 74, 131–132, 148–150, 193, 196, 320–321
 - Default Instance Security settings 193
 - default owner settings 199
 - real time lookup 79
- document correction 77
- Document instance 151, 196, 199–201
- document instance
 - access control 226
 - particular permissions 200
 - view content right 203
- document object 182
- Document Type Definition (DTD) 153
- document type recognition 77
- DOCVERSION table 36
- dynamic permission 219
- dynamic security inheritance 213
- Dynamic Security Inheritance Object, 211

E

- ECM system
 - critical building block 149
- eDiscovery Analyzer 15, 57, 134
- eDiscovery Manager 15, 57, 63–64, 130, 134, 139
 - integration point 135
 - integration with IBM FileNet Records Manager 136
 - system 134
 - user interface 136
- eForms 47, 109–110, 112–114, 168–169
 - key features 110
 - scaling 281
- EJB 99
- EJB listener 39

- EJB transport 231, 238, 250, 261, 267
 - Content Engine 284
- Electronic form
 - key features 110
- electronic form 2, 10, 108–109, 167–168, 245, 281, 312, 314, 319
- Electronic forms (eForms) 57
- Email Manager 50, 135
- email notification 18
- email system 2, 315
- emails 49
- enterprise content management (ECM) xvii, 1, 4, 6
- Enterprise Manager 34, 37, 39, 234
- Enterprise Portal
 - Repository Framework 285
- Enterprise Portal (EP) 83–84, 281, 284–285
- Enterprise Reference Architecture (ERA) 23, 25
- ERA
 - service layers 24
- ERP 86
- event 155
 - subscription 159
- event action 33–34
 - handler 33–34
- Event Activator 74
- event data 127
- event framework 33
- event log 41, 45
- events 125
- expansion product 55, 80–82, 107, 123, 129, 196, 209, 225
 - scaling 280
- expansion products 56
- external system 124, 141, 159–160, 170, 223, 281, 299, 315
 - integration 170
 - interactions 170
 - operations 170
 - response 160
 - web service 171

F

- failover mechanism 328
- farm
 - Content Engine 36
 - Process Engine servers 277
- federated access 147
- federated content 146

- federated content query 90
- federated records management 184
- federation
 - content 145
 - search 146
- file plan 52, 75
- File Plan Object Store (FPOS) 51, 182
- file server (FS) 325
- file share 325
- file storage area 269
- firewall 305
- Fixed Content Device (FCD) 28, 164, 269–271
- Fixed Content Device (FCD) storage 35–36
- fixed content services 35
- fixed storage area 270
- Folder object 31
- form template 168
- Forms Integration Framework 169
- full-text indexing 36
- fulltext indexing 272
- full-text search 20
- fulltext search 272, 274
 - scaling 274

G

- generational concurrent (GC) 292–293
- global configuration database (GCD) 30, 293, 300
- GUID 31

H

- HA cluster 307
- Hardware load balancer 256, 258
 - Content Engine 267
- high availability 312–313
 - single site implementation 313
- high availability (HA) 307–308
- horizontal scaling 251–252, 323
- http plug-in 269

I

- IBM Classification Module 57
 - architecture 132
- IBM Classification Module (ICM) 14, 60, 130, 162, 184, 318, 325
- IBM Cognos 124
- IBM Content Collector 56, 247
 - application integration 65

- Configuration Manager 59
 - main components 62
 - scaling 286
 - system architecture 62
- IBM Content Collector (ICC) 11, 58, 135, 163, 286–287
- IBM Content Collector for Email 58
- IBM Content Collector for File Systems 58
- IBM Content Integrator 56
- IBM FileNet
 - APIs 18
 - Business Activity Monitor 46, 289
 - Business Process Framework 114, 239, 247
 - Business Process Manager 7, 9, 28, 112, 124, 134, 209, 220, 223
 - Capture 58–59, 67
 - Capture ADR 77
 - Capture Advanced Document Recognition (ADR) 75
 - Capture Desktop 71
 - Capture Path 76
 - Capture product 70
 - Capture Professional 67, 70
 - Capture technology 69
 - Connectors 56, 63
 - Content Manager 7, 9, 28, 50, 54, 61, 64, 84–85, 112, 136, 206, 209
 - Content Manager suite 50
 - content repository 101
 - Content Service 79
 - data model 92
 - ECM prototype 91
 - ECM solution 12
 - eForms 47, 112–113
 - Email Manager 12, 50, 58–59, 135
 - Enterprise Manager 34, 37, 39, 234
 - expansion products 94
 - Fax 71, 78
 - front-end application 83
 - Image Manager repository 35
 - Image Service 86
 - Image Services 61, 79, 255, 269, 283
 - Image Services repository 22
 - P8-based compliance 67
 - P8-driven product 9
 - product family 16
 - Records Crawler 12, 58
 - Records Management 94
 - Records management solution 183
 - Records Manager 7, 10, 15, 28–29, 50, 60, 79, 130, 139, 151, 180, 208–209
 - Records Manager application 21
 - Records Manager architecture 51
 - Records Manager implementation 209
 - Records Manager Java API 185
 - Records Manager reporting mechanism 239
 - Remote Capture 78
 - Remote Capture Services 78
 - repository 79, 83, 92
 - server 74
 - services 78
 - System Monitor 16
 - Web Application Toolkit 19
 - Workplace
 - user interface 86
- IBM FileNet Application Connector for SAP R/3 56
- IBM FileNet Capture 56, 83, 285
- IBM FileNet Capture ADR 77
- IBM FileNet Capture Professional 72–73, 79
- IBM FileNet Connectors for Microsoft SharePoint 57
- IBM FileNet Content Manager
 - implementation 149
- IBM FileNet Fax 79
- IBM FileNet P8 48, 82, 85
 - applications leverage 6
 - architecture 8, 16, 51, 56, 141–142, 311–313
 - Catalog Management Service 21
 - component 22
 - content 47–49
 - Content Engine 33
 - Content Engine (CE) 29
 - content management 88
 - Content Manager 75, 324
 - content repository 10, 47
 - core components 29
 - Enterprise Reference Architecture 25
 - environment 187
 - expansion product 56
 - expansion products 56
 - functionality 88
 - hardware 33
 - object store 49, 104
 - Platform 1, 6–7, 16, 27–28, 56–58, 67, 82, 88, 100, 114, 122, 139, 141, 170, 187, 208, 249
 - Platform component 28
 - Platform, performance 291
 - portfolio 10, 58, 255

- product 1, 9, 187
- products share 17
- repository 11, 13, 65, 68, 80, 104
- Repository Manager 285
- solution 8
- system 56–57, 60, 106, 139, 180
- system, distributed 295
- system, DMZ deployment 305
- IBM FileNet Services for Lotus Quickr 56
- IBM OmniFind Enterprise Edition 57
- ICC cluster 286
- ICM system 325
- Image Manager
 - repository 35
- Image Services
 - architecture 289
 - scaling 284
- Image Services Resource Adapter (ISRA) 283
- image verification 73
- inbasket 115–116, 195, 227, 314
- inbox 104–105
- incoming request 40, 233, 251–252
 - huge number 256
- index skew 294
- Index Verify 74
- indexing 74
 - fulltext 272
 - fulltext, scaling 272
- indexing proces 37
- indexing service 37
- Information Lifecycle Management
 - key component 25
- Information Lifecycle Management (ILM) 25
- ingestion
 - content 132
- inheritance 148
 - dynamic security inheritance 213
- iNotes 66
- Intelligent Character Recognition (ICR) 76
- invoke step 171
- ISIS 71
- isolated region 40–41, 142, 194, 260, 276
 - individual BPF applications 280
 - small number 40
 - work objects 145

J

- J2EE application 29

- JAAS 189, 250
- JAAS context 188, 190
- Java API 183, 185, 188, 195, 261, 265
 - Records Manager 185
 - WSI transport 267
- Java Authentication and Authorization Service (JaaS) 32, 48
- Java class 159, 164, 195, 212
- Java Content Repository (JCR) 99
- Java Messaging Queues (JMS) 170
- Java Messaging System (JMS) 260–261
- Java Naming Directory Interface (JNDI) 257
- Java Runtime Environment (JRE) 172, 257
- Java Transaction API
 - client transactions base 261
- Java Transaction API (JTA) 261
- Java Virtual Machine (JVM) 90, 292
 - performance 292
- JMS queue 50

K

- K2 server 37
- Kerberos 190
- Knowledge Base 131
- knowledge base (KB) 16, 131
- knowledge management (KM) 5, 8, 84–85
- Kofax 71

L

- large scale system 326
- legal hold 147
- legal review 134
- lifecycle 147
- lifecycle action 34, 163–164, 212, 235
- lifecycle actions 34
- lifecycle polices 34
- lifecycle policy 34, 163–164, 202, 204, 212
 - document lifecycle management 34
- Lightweight Directory Access Protocol (LDAP) 188
- Link Objects 153
- Link Objects class 152
- load balancer 230, 255–256, 269
 - RMI-IIOP requests 257
 - session affinity 268
 - virtual address 262
- load balancing 101, 251, 255–256
- load modelling 327
- local area network (LAN) 302

- logging 233, 236
 - Content Engine 234
 - Process Engine 235

- Lotus Form 109, 112

- Lotus forms 169

- Lotus Quickr 91

- connector 96

- EJB 97

- IBM FileNet Services 94, 97

- place 93

- team collaboration 94

- LPAR 307

M

- Major Versioning

- permission 193

- marking 205

- marking set 205, 211

- types 206

- marking sets 217

- merge component 74

- metadata 144

- common classification 163

- content objects 144

- metropolitan area network (MAN) 302

- Microsoft Office

- client 229

- integration 229

- SharePoint Server 13

- Microsoft Outlook 49–50, 61, 63, 93

- Microsoft Outlook

- offline access 65

- outlook extension 65

- Microsoft SharePoint 13, 91

- activity 105

- automated tasks 104

- connector architecture 102

- content 13

- Document Libraries 101

- implementation 101

- product 101

- monitoring 174

- monitoring dashboard 176

- Multi Function Printer/Device (MFP) 78

- multi key file (MKF) 290

N

- Network Deployment (ND) 257, 265

- Network File System (NFS) 35

- network layer 329

- network-attached storage (NAS) 35

O

- object class 149

- Object Management Group (OMG) 250

- object request broker (ORB) 250, 257

- object store 22, 30, 39, 58, 104, 132, 138, 150, 200, 216, 219, 271

- contained object 203

- database 157, 269–270

- owner right 200

- valid objects 215

- OCR2PDF 73, 75

- OLAP cube 46, 142, 179, 288

- OmniFind Enterprise Edition

- architecture 137

- Optical Character Recognition (OCR) 69, 74

- Optical Mark Recognition (OMR) 76–77

- optimize

- business process 128

- Outlook Web Access (OWA) 65

- owner

- permission 200

P

- P8 components

- securing 192

- paravirtualization 255

- patch code 73

- peak time 316

- performance 22

- application design 294

- database indexes 293

- IBM FileNet P8 Platform 291

- permission 215

- Major Versioning 193

- owner 200

- View Content 193

- view content 193, 203–204

- permissions 197, 242

- PEWS 262

- platform 6

- polling time 125

- portal 196

- precedence

- security 202

- privileged group 217
- process
 - optimize 128
- Process Analyzer 127
 - workflow statistics 127
- Process Analyzer (PA) 18, 105, 124, 126, 178, 237, 288, 318
 - horizontal scaling 289
- Process Configuration Console 236
- process definition 18, 40–41, 142, 166, 173, 289
- Process Definition Language (XPDL) 10
- Process Engine 260
 - audit capabilities 158
 - business activity monitoring 178
 - corresponding containment model 40
 - event log configuration 237
 - farm 277
 - features 166
 - integration 174
 - logging 235
 - production event data 127
 - scaling 276
 - scaling options 23
- Process Engine (PE) 7, 9, 17, 28–29, 40, 111, 154, 187, 194, 220, 227, 250, 257, 316
 - data model 41
 - database 41
 - database schema 43
 - queues 42
 - system architecture 40
- process instance 47, 168, 170, 224
- Process orchestration 44, 50, 171–172, 194, 260–261
- process queue 42
- Process Simulator 18, 127
- Process Task Manager 18
- Process Web Services 18
- properties
 - multi-valued 31
- property 31
- property type 31
- Web services provider 44
- proxied object 33
- proxy server 259
- Public Records Office (PRO) 52
- publishing service 38
- publish-subscribe schema 155

Q

- queue 41
- queue element 43
- queue filter 122
- queues 42

R

- read only (RO) 133
- Read/Write (RW) 133
- record folder 52–53, 183
- Record Information Object (RIO) 51–53
- Record Information Objects (RIO) 54
- record object 182
- record review 225
- Record Volume 52
- Records Activator 73, 75
- Records Administrator 52
- records hold 184
- records management 6, 52, 75, 88, 94, 130, 137, 141, 146–147, 151, 181, 246
 - electronic messaging data stores 12
 - key factors 181
- Records Manager 28–29, 50, 60, 79, 130, 139, 190, 208, 225, 265, 267
 - capture 79
 - object 185
 - web application 181, 185
- Records Manager (RM) 10, 15
- Records Manager Java API 185
- Records-enabled Object Store (ROS) 51, 182
 - common records schema 182
- Redbooks Web site 332
 - Contact us xx
- Referential Containment Relationship 31
- Relational Database Management System 2, 28, 35
- Remote Capture 71
- remote location 297, 299, 323
 - Content Engine servers 297
 - limited bandwidth 299
 - work items 300
- Remote Method Invocation (RMI) 90, 250
- remote site 296, 298
 - Application Engine 299
- reply step 171
- repository
 - Image Manager 35
- Repository Manager 85
- Repository Manager (RM) 285

- request 233
- REST protocol 99
- retention period 50, 184, 247, 270
- retention schedule 52
- retrieve
 - content management 143
- RightFax Enterprise Server 78
- RMI-IIOP 250
- role 115
- roster element 43
- rules connectivity framework 18
- rules engine 46
- Rules Engine Framework 174

S

- scalability 22
- scaling 279
 - ACSAP for R/3 283
 - add-on products 280
 - Business Process Framework 280
 - Capture 285
 - Content Search Engine 272
 - eForms 281
 - expansion products 280
 - horizontal 251–252, 289, 323
 - IBM Content Collector 286
 - Image Services 284
 - vertical 251–252
- scanning
 - bulk 314
- search 104
- Search and Index API (SI-API) 138
- search federation 146
- search result 105
- search template 185
- search templates
 - Content Engine 185
- Secure Sockets Layer (SSL) 230
- securing
 - P8 components 192
- security 147, 227
 - authorization 203
 - calculation 202
 - Content Engine 105, 146
 - default instance security 199
 - dynamic security inheritance 213
 - manage update 220, 226
 - update in real time 220

- view content permission 193, 203–204
- workplaceXT access roles 195
- security access rights 39
- security folder 241
- Security Folder property 215
- security methods
 - advantages and disadvantages 221
- security policy 102, 164, 193, 202, 209, 321
- security rights 44
- security setting 21, 33, 202–203, 216
- security templates
 - Content Engine 211
- segmentation 77
- servers support token (SSO) 189–190
- service level agreement (SLA) 16, 143, 167, 176, 238, 314, 318
- Service Oriented Architecture 324
- Service-oriented architecture (SOA) 19
- session affinity 255
- session awareness 255
- session stickiness 255
- shared file system
 - temporary location 37
- shared service 218
- SharePoint Document Libraries
 - connector 101
- Simple Object Access Protocol (SOAP) 92
- single sign on (SSO) 105
- SOAP 78
- solution template 313, 319
- SSL 230
- SSO
 - authentication 191
 - mechanism 190
- stateless
 - Content Engine 36
- statistics collection 294
- step element 43
- storage
 - cost 314
- storage area 269–270
- storage options
 - Content Engine 35
- storage services
 - Content Engine 36
- store
 - content management 143
- stored workflow definition
 - workflow information 127

- subscriptions
 - event 159
- system architecture
 - Application Engine 48
 - Content Engine 30
 - IBM Content Collector 62
 - Process Engine 40
- system queue 42

T

- task connector 60
- Task Manager 261
- task route 59
- taxonomy 147, 161
- Taxonomy Proposer 163
- templates
 - search, Content Engine 185
 - security, Content Engine 211
- Tivoli Performance Monitoring (TPM) 293
- transport layer 99
- Twain 71

U

- UFI Task Route Engine 103
- UFI Utility Service 104
- Universal Description, Discovery and Integration (UDDI) 258
- Universal Resource Name (URN) 89
- Unstructured Information Management Architecture (UIMA) 15
- user interface 60, 72
- user queue 42

V

- vertical scaling 251–252
- view content
 - permission 193, 203–204
 - permission entry 205
- View Content permission 193
- virtualization 254
- virtualized environment 142, 253–254
- VWKs processes 40

W

- Web application login box 190
- web application 58, 60, 63, 112, 123, 155, 185, 190, 229, 257, 261

- Web Application Server 85–86
- Web Application Toolkit 19
- Web service
 - WS-Security credentials 232
- web service 39, 44, 91, 97, 171, 231, 233, 250, 257, 315, 325
 - NET API 39
- Web Services
 - Business Process Execution Language 171
- Web Services Framework 19
- web services queue 261
- WebSphere Business Monitor (WBM) 175, 177
- WebSphere Process Server (WPS) 177
- wide area network (WAN) 8, 296
- wide range 28
- Windows SharePoint Services (WSS) 13
- work item 21, 42, 44, 89, 120, 122, 144, 194, 227, 262, 300
 - corresponding properties 45
 - event action 45
 - lock down 220
 - various system activities 42
- work object 43, 84, 142, 144–145, 168, 276
 - attachment fields 168
- work request 260
- Workbench 125
- workflow definition 161, 169, 194
 - version 161
- workflow form policy 168
- workflow group 224
- workflow inbox 13
- workflow item 227
- workflow statistics 127
- Workflow Subscription 111, 145, 160
- working option 122
- workplace
 - access roles 195
- WorkplaceXT 22, 39, 110–111, 153, 160, 191, 195, 259, 262
 - access roles 195
- Write Once Read Many (WORM) 35
- WSDL 78
- WSDL lookup 258
- WSI transport 250, 261
- WSRequest 261

Z

- ZeroClick 183



IBM FileNet P8 Platform and Architecture

(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages



IBM FileNet P8 Platform and Architecture

**Architecture and
expansion products**

**Enterprise content
management**

**Scalability and
distribution**

IBM FileNet P8 Platform is a next-generation, unified enterprise foundation for the integrated IBM FileNet P8 products. It combines the enterprise content management with comprehensive business process management and compliance capabilities. IBM FileNet P8 addresses the most demanding compliance, content, and process management needs for your entire organization. It is a key element in creating an agile, adaptable enterprise content management (ECM) environment necessary to support a dynamic organization that must respond quickly to change.

In this IBM Redbooks publication, we provide an overview of IBM FileNet P8 and describe the core component architecture. We also introduce major expansion products that extend IBM FileNet P8 functionality in the areas of content ingestion, content accessing through connectors and federation, the application framework, and discovery and compliance. In this book, we discuss the anatomy of an ECM infrastructure, content event processing, content life cycle, and business processes.

We designed this book to give IT architects, IT specialists, and IT Technical Sales a solid understanding of IBM FileNet P8 Platform, its architecture, its functions and extensibility, and its unlimited capabilities.

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