Actuate One™

One Design
One Server
One User Experience

Using BIRT iHub Integration Technology

Information in this document is subject to change without notice. Examples provided are fictitious. No part of this document may be reproduced or transmitted in any form, or by any means, electronic or mechanical, for any purpose, in whole or in part, without the express written permission of Actuate Corporation.

© 1995 - 2013 by Actuate Corporation. All rights reserved. Printed in the United States of America.

Contains information proprietary to:
Actuate Corporation, 951 Mariners Island Boulevard, San Mateo, CA 94404

www.actuate.com

The software described in this manual is provided by Actuate Corporation under an Actuate License agreement. The software may be used only in accordance with the terms of the agreement. Actuate software products are protected by U.S. and International patents and patents pending. For a current list of patents, please see http://www.actuate.com/patents.

Actuate Corporation trademarks and registered trademarks include:
Actuate, ActuateOne, the Actuate logo, Archived Data Analytics, BIRT, BIRT 360, BIRT Analytics, The BIRT Company, BIRT Data Analyzer, BIRT iHub, BIRT Performance Analytics, Collaborative Reporting Architecture, e.Analysis, e.Report, e.Reporting, e.Spreadsheet, Encyclopedia, Interactive Viewing, OnPerformance, The people behind BIRT, Performancesoft, Performancesoft Track, Performancesoft Views, Report Encyclopedia, Reportlet, X2BIRT, and XML reports.

Actuate products may contain third-party products or technologies. Third-party trademarks or registered trademarks of their respective owners, companies, or organizations include:
Mark Adler and Jean-loup Gailly (www.zlib.net): zLib. Adobe Systems Incorporated: Flash Player. Amazon Web Services, Incorporated: Amazon Web Services SDK, licensed under the Apache Public License (APL). Apache Software Foundation (www.apache.org): Ant, Axis, Axis2, Batik, Batik SVG library, Commons Command Line Interface (CLI), Commons Codec, Crimson, Derby, Hive driver for Hadoop, Pluto, Portals, Shindig, Struts, Tomcat, Xalan, Xerces, Xerces2 Java Parser, and Xerces-C++ XML Parser. Castor (www.castor.org), ExoLab Project (www.exolab.org), and Intalio, Inc. (www.intalio.org): Castor. Day Management AG: Content Repository for Java. Eclipse Foundation, Inc. (www.eclipse.org): Babel, Data Tools Platform (DTP) ODA, Eclipse SDK, Graphics Editor Framework (GEF), Eclipse Modeling Framework (EMF), and Eclipse Web Tools Platform (WTP), licensed under the Eclipse Public License (EPL). Gargoyle Software Inc.: HtmlUnit, licensed under Apache License Version 2.0. GNU Project: GNU Regular Expression, licensed under the GNU Lesser General Public License (LGPLv3). HighSlide: HighCharts. Jason Hsueth and Kenton Varda (code.google.com): Protocole Buffer. IDAutomation.com, Inc.: IDAutomation. IDRsolutions Ltd.: JBIG2, licensed under the BSD license. InfoSoft Global (P) Ltd.: FusionCharts, FusionMaps, FusionWidgets, PowerCharts. Matt Inger (sourceforge.net): Ant-Contrib, licensed under Apache License Version 2.0. Matt Ingenthron, Eric D. Lambert, and Dustin Sallings (code.google.com): Spymemcached, licensed under the MIT OSI License. International Components for Unicode (ICU): ICU library. jQuery: jQuery, licensed under the MIT License. Yuri Kanivets (code.google.com): Android Wheel gadget, licensed under the Apache Public License (APL). LEAD Technologies, Inc.: LEADTOOLS. The Legion of the Bouncy Castle: Bouncy Castle Crypto APIs. Bruno Lowagie and Paulo Soares: iText, licensed under the Mozilla Public License (MPL). Microsoft Corporation (Microsoft Developer Network): CompoundDocument Library. Mozilla: Mozilla XML Parser, licensed under the Mozilla Public License (MPL). MySQL Americas, Inc.: MySQL Connector. Netscape Communications Corporation, Inc.: Rhino, licensed under the Netscape Public License (NPL). OOPS Consultancy: XMLTask, licensed under the Apache License, Version 2.0. Oracle Corporation: Berkeley DB, Java Advanced Imaging, JAXB, JDK, Jstl. PostgreSQL Global Development Group: pgAdmin, PostgreSQL, PostgreSQL JDBC driver. Progress Software Corporation: DataDirect Connect XE for JDBC Salesforce, DataDirect JDBC, DataDirect ODBC. Rogue Wave Software, Inc.: Rogue Wave Library SourcePro Core, tools.h++. Sam Stephenson (prototype.conio.net): prototype.js, licensed under the MIT license. Sencha Inc.: Ext JS, Sencha Touch. ThimbleWare, Inc.: JMemcached, licensed under the Apache Public License (APL). World Wide Web Consortium (W3C) (MIT, ERCIM, Keio): Flute, JTidy, Simple API for CSS. XFree86 Project, Inc.: (www.xfree86.org): xvfb. ZXing authors (code.google.com): ZXing, licensed under the Apache Public License (APL).

All other brand or product names are trademarks or registered trademarks of their respective owners, companies, or organizations.

Document No. 130131-2-430301 January 23, 2013

Contents

About *Using BIRT iHub Integration Technology* xvii

Part 1
Introduction to the Actuate Information Delivery API

Chapter 1
Understanding the Information Delivery API and schema 3
About the Actuate Information Delivery API 4
About web services and WSDL 5
Understanding the elements of iHub's WSDL schema 5
About the definitions element 6
About data type definitions 7
About message definitions 9
About the portType definition 9
About the binding definition 9
About the service definition 10
Accessing the Actuate schema using a web browser 11

Chapter 2
Constructing a SOAP message 13
About SOAP messaging 14
Calling an Actuate web service 14
About SOAP message elements 15
Understanding the HTTP header 16
Understanding the SOAP envelope 17
About XML namespace declarations 18
Understanding the SOAP header 19
Understanding the SOAP message body 21
About SOAP Fault messages 22

Chapter 3
Understanding Actuate Information Delivery API operations 25
About Actuate Information Delivery API operations 26
Working with BIRT iHub 26
Working with channels 29
Working with Encyclopedia volumes 29
Working with files and folders 31
Working with groups 34
Working with information objects and databases 35

Working with jobs and reports 35
Working with searches 40
Performing a search operation 40
Using special characters in searches 41
Working with users, roles, and security 42

Chapter 4
Running, printing, and viewing a document 45
Generating or printing a document 46
Running a synchronous report 46
Setting a time frame for an ExecuteReport response 48
Waiting for report generation 49
Running a synchronous report that uses parameters 50
Retrieving report parameter values 51
Running or printing a job 52
Understanding SubmitJob 53
Specifying parameters for a job 54
About hidden, required parameters 54
Scheduling report generation or printing 54
Working with a job notification 56
About e-mail attachments 56
About notifying a channel 57
Sending an e-mail notification using SubmitJob 57
Sending an e-mail notification using UpdateJobSchedule 58
Notifying a channel using SubmitJob 59
Notifying a channel using UpdateJobSchedule 60
Customizing an e-mail notification 61
Using the e-mail template for multiple locales 61
Retrieving job properties using GetJobDetails 61
Retrieving job properties using GetNoticeJobDetails 63
Cancelling a job 64
Working with a resource group 65
Creating an asynchronous resource group 66
Creating a synchronous resource group 67
Updating a resource group's properties 68
Getting a list of resource groups 69
Retrieving the properties of a specific resource group 70
Retrieving properties for all resource groups on a BIRT iHub 70
Setting properties for the resource groups on a BIRT iHub 71
Deleting a resource group 73
Assigning a report to a resource group 73
Assigning a job to a resource group 74
Retrieving the resource group to which a job is assigned 75

Retrieving and viewing data 76
Requesting a page or range of pages using SelectPage 77
Retrieving the attachment to a SelectPage response 78
Using SelectPage to print 78
Retrieving report content 79
Retrieving embedded data and style sheets 82
Searching within a document 82
Searching for a range of pages using SearchReport 84
Getting a table of contents 85
Requesting a page count 87
Retrieving display formats 87
Retrieving a custom format 88
Managing a large list 89
Working with a large message 90
Delivering a multilingual document 93

Chapter 5
Administering an Encyclopedia volume 95
About the Encyclopedia service 96
Defining the data on which an operation acts 96
Defining data using Id or IdList 96
Defining data using Name or NameList 97
Defining data using Search 97
Administering security and authentication operations 98
Logging in as a report user 99
Logging in with SystemLogin 101
Getting an access control list 101
Requesting a file or folder's ACL 101
Retrieving the ACL for a channel 102
Getting a user's ACL template 103
Setting an additional condition on an ACL request 104
About Encyclopedia-level management operations 105
Uploading a file 105
Uploading a BIRT design 106
Uploading a third-party report 107
Copying file properties when uploading a file 107
Attaching or embedding a file in a request 108
Uploading a file as an attachment 109
About the HTTP header 109
Writing the UploadFile request 110
Downloading a file 111
Downloading a file as an attachment 112
Updating a file or folder 112

Updating a file's parameters . . . 113
Updating the privilege settings of a file or folder . . . 114
Selecting properties of a file or folder in an Encyclopedia volume . . . 116
Requesting a list of files or folders in a working directory using SelectFiles . . . 117
About the Search element in SelectFiles . . . 118
Using a privilege filter with SelectFiles . . . 119
Retrieving a property list for an item in a folder . . . 120
Retrieving properties of an item in a folder . . . 121
Setting a condition using GetFolderItems . . . 122
Using GetFileDetails . . . 123
Getting the details of an Actuate report . . . 123
Managing Encyclopedia volume items . . . 124
Ignoring error conditions in an Administrate operation . . . 124
Creating an item in an Encyclopedia volume . . . 124
Creating a user . . . 125
Creating a folder . . . 125
Creating a security role . . . 126
Deleting an item . . . 127
Deleting a file or folder . . . 127
Deleting a user . . . 128
Updating an item . . . 128
Updating a job schedule . . . 129
Updating a channel . . . 130
Moving a file or folder . . . 131
Copying a file or folder . . . 131
About composite operations and transactions . . . 132
About sequences in composite Administrate operations . . . 133
Working with a transaction . . . 133
About TransactionOperation and AdminOperation . . . 135
Searching within an Encyclopedia volume . . . 136
Selecting an item in an Encyclopedia volume . . . 136
Selecting a job or job list . . . 137
Getting Encyclopedia volume, printer, and file type information . . . 138
Getting Encyclopedia volume properties . . . 139
Getting BIRT iHub System printer information . . . 140
Retrieving a user's printer settings . . . 142
Retrieving parameter definitions for a file type . . . 142
Extracting parameter definitions . . . 143
Exporting file parameters . . . 144
Executing a predefined Encyclopedia volume command . . . 145
Diagnosing reporting environment problems . . . 146
About Ping request options . . . 146

Sending a Ping request in Concise mode 149
Sending a Ping request in Normal mode 149
Sending a Ping request in Trace mode 150
Monitoring BIRT iHub information 151
Getting information about BIRT iHub 151
Getting information about a running or pending job 152
Getting information about Factory service processes 155
Monitoring or cancelling a request for a synchronous report 156
Monitoring a request for a synchronous report 156
Cancelling a request for a synchronous report 157

Part 2
Developing Actuate Information Delivery API applications

Chapter 6
Developing Actuate Information Delivery API applications using Java 161
About the Apache Axis 1.4 client 162
Generating the com.actuate.schemas library 162
About third-party code libraries 163
About the Actuate Information Delivery API framework 164
Using a data type from a WSDL document to generate a JavaBean 165
Using metadata to map XML to a Java type 166
Mapping the portType to a Service Definition Interface 167
Using a WSDL binding to generate a Java stub 168
Implementing the Actuate API service 169
Developing Actuate Information Delivery API applications 170
Writing a program that logs in to an Encyclopedia volume 171
About the auxiliary classes provided by the sample application 173
Logging in to the Encyclopedia volume 176
Capturing SOAP messages using Axis TCPMonitor 177
Writing a simple administration application 178
Creating a user 179
About ActuateControl.createUser( ) 180
About ActuateControl.runAdminOperation( ) 181
Performing a search operation 182
Using com.actuate.schemas.SelectFiles 182
Using ResultDef 185
Writing a batch or transaction application 187
About batch and transaction operations 187
Implementing a transaction-based application 188

Uploading a file 190
About ways of uploading a file 190
Using com.actuate.schemas.UploadFile 191
How to build an application that uploads a file 191
Downloading a file 194
Using com.actuate.schemas.DownloadFile 195
How to build an application that downloads a file 195
Executing a report 198
Understanding the structure of an ExecuteReport application 199
Using SelectJavaReportPage 200
Scheduling a custom event 206
Implementing a custom event service 209
Building a custom event service 210
About the custom event web service sample 210
SOAP-based event web service operations and data types 212
ArrayOfEvent 212
ArrayOfEventStatus 212
Event 212
EventStatus 213
GetEventStatus 213

Chapter 7
Developing Actuate Information Delivery API applications using Microsoft .NET 215
About the Microsoft .NET client 216
About the Actuate Information Delivery API framework 219
Using a data type from a WSDL document to generate a C# class 219
Mapping the portType to a web service interface 221
Developing Actuate Information Delivery API applications 222
Writing a program that logs in to an Encyclopedia volume 222
Writing a simple administration application 226
Performing a search operation 227
Writing a batch or transaction application 230
About batch and transaction operations 231
Implementing a transaction-based application 231
Uploading a file 233
About ways of uploading a file 233
Using UploadFile 234
How to build an application that uploads a file 234
Downloading a file 235
Using DownloadFile 236
How to build an application that downloads a file 236

Chapter 8

Actuate Information Delivery API operations 239
About the SOAP header 240
Administrate 241
AdminOperation 241
CallOpenSecurityLibrary 244
CancelJob 245
CancelReport 246
CloseInfoObject 246
CopyFile 247
CreateChannel 249
CreateDatabaseConnection 249
CreateFileType 250
CreateFolder 250
CreateGroup 251
CreateParameterValuesFile 252
CreateResourceGroup 253
CreateRole 253
CreateUser 254
DataExtraction 254
DeleteChannel 256
DeleteDatabaseConnection 257
DeleteFile 257
DeleteFileType 258
DeleteGroup 259
DeleteJob 260
DeleteJobNotices 261
DeleteJobSchedule 261
DeleteResourceGroup 262
DeleteRole 262
DeleteUser 263
DownloadFile 264
DownloadTransientFile 265
ExecuteQuery 266
ExecuteReport 269
ExecuteVolumeCommand 271
ExtractParameterDefinitionsFromFile 273
ExportParameterDefinitionsToFile 273
FetchInfoObjectData 274
GetBookmarks 275
GetChannelACL 275
GetConnectionPropertyAssignees 277

GetContent 278
GetCustomFormat 280
GetDatabaseConnectionDefinition 281
GetDatabaseConnectionParameters 281
GetDatabaseConnectionTypes 282
GetDataExtractionFormats 283
GetDocumentConversionOptions 283
GetDynamicData 284
GetEmbeddedComponent 285
GetFactoryServiceInfo 288
GetFactoryServiceJobs 289
GetFileACL 292
GetFileCreationACL 294
GetFileDetails 296
GetFileTypeParameterDefinitions 297
GetFolderItems 297
GetFormats 299
GetInfoObject 299
GetJavaReportEmbeddedComponent 300
GetJavaReportTOC 301
GetJobDetails 302
GetMetaData 305
GetNoticeJobDetails 306
GetPageCount 309
GetPageNumber 310
GetParameterPickList 310
GetReportParameters 312
GetResourceGroupInfo 313
GetResourceGroupList 314
GetSavedSearch 315
GetServerResourceGroupConfiguration 315
GetStaticData 316
GetStyleSheet 317
GetSyncJobInfo 318
GetSystemMDSInfo 319
GetSystemPrinters 320
GetSystemServerList 320
GetSystemVolumeNames 321
GetTOC 322
GetUserLicenseOptions 323
GetUserPrinterOptions 324
GetVolumeProperties 325
Login 326

MoveFile . . . 328
ODBOTunnel . . . 330
OpenInfoObject . . . 331
Ping . . . 332
PrintReport . . . 337
SaveSearch . . . 339
SaveTransientReport . . . 340
SearchReport . . . 341
SelectChannels . . . 343
SelectFiles . . . 344
SelectFileTypes . . . 346
SelectGroups . . . 347
SelectJavaReportPage . . . 348
SelectJobs . . . 350
SelectJobNotices . . . 351
SelectJobSchedules . . . 353
SelectPage . . . 354
SelectRoles . . . 356
SelectUsers . . . 358
SetConnectionProperties . . . 360
SetServerResourceGroupConfiguration . . . 361
SubmitJob . . . 361
SystemLogin . . . 368
Transaction . . . 369
TransactionOperation . . . 369
UndeleteUser . . . 372
UpdateChannel . . . 373
UpdateChannelOperation . . . 374
UpdateChannelOperationGroup . . . 376
UpdateDatabaseConnection . . . 376
UpdateFile . . . 377
UpdateFileOperation . . . 378
UpdateFileOperationGroup . . . 381
UpdateFileType . . . 381
UpdateFileTypeOperation . . . 382
UpdateFileTypeOperationGroup . . . 383
UpdateGroup . . . 383
UpdateGroupOperation . . . 384
UpdateGroupOperationGroup . . . 385
UpdateJobSchedule . . . 386
UpdateJobScheduleOperation . . . 387
UpdateJobScheduleOperationGroup . . . 391
UpdateOpenSecurityCache . . . 391

UpdateResourceGroup 392
UpdateRole 393
UpdateRoleOperation 394
UpdateRoleOperationGroup 396
UpdateUser 397
UpdateUserOperation 398
UpdateUserOperationGroup 401
UpdateVolumeProperties 402
UpdateVolumePropertiesOperation 402
UpdateVolumePropertiesOperationGroup 403
UploadFile 404
WaitForExecuteReport 405

Chapter 9
Actuate Information Delivery API data types 407
AbsoluteDate 408
acDouble 408
acNull 408
Aggregation 409
ArchiveRule 409
Argument 410
Arrays of data types 411
Attachment 412
BookMark 413
Boolean 414
CancelJobStatus 414
Channel 414
ChannelCondition 415
ChannelField 416
ChannelSearch 416
ColumnDefinition 418
ColumnDetail 420
ColumnSchema 421
ComponentIdentifier 422
ComponentType 422
CustomEvent 423
Daily 423
DatabaseConnectionDefinition 424
DataCell 425
DataExtractionFormat 425
DataFilterCondition 426
DataRow 426
DataSchema 427

DataSortColumn 427
DataSourceType 427
DataType 428
DocumentConversionOptions 429
Event 429
EventOptions 431
EventType 431
ExecuteReportStatus 432
ExternalTranslatedRoleNames 433
FieldDefinition 433
FieldValue 435
File 435
FileAccess 437
FileCondition 437
FileContent 438
FileEvent 438
FileField 438
FileSearch 439
FileType 441
FilterCriteria 443
FormatType 444
Group 444
GroupCondition 445
GroupField 445
Grouping 446
GroupSearch 446
Header 448
Integer 449
InfoObjectData 449
InfoObjectDataFormat 449
JobCondition 450
JobEvent 450
JobField 451
JobInputDetail 452
JobNotice 457
JobNoticeCondition 459
JobNoticeField 460
JobNoticeSearch 460
JobPrinterOptions 461
JobProperties 463
JobSchedule 467
JobScheduleCondition 467
JobScheduleDetail 468

JobScheduleField 469
JobScheduleSearch 470
JobSearch 472
LicenseOption 474
MDSInfo 474
Monthly 475
NameValuePair 477
NewFile 477
ObjectIdentifier 479
OpenServerOptions 479
PageIdentifier 480
ParameterDefinition 480
ParameterValue 484
PendingSyncJob 485
Permission 487
Printer 488
PrinterOptions 491
PrivilegeFilter 493
PropertyValue 493
Range 494
Repeat 494
Record 494
ReportParameterType 495
ResourceGroup 495
ResourceGroupSettings 497
ResultSetSchema 498
RetryOptions 498
RetryOptionType 499
Role 499
RoleCondition 500
RoleField 500
RoleSearch 500
RunningJob 502
ScalarDataType 504
SearchReportByIdList 505
SearchReportByIdNameList 506
SearchReportByNameList 506
SearchResultProperty 507
ServerInformation 507
ServerResourceGroupSetting 509
ServerState 510
ServerStatusInformation 510
ServerVersionInformation 511

Service 512
SortColumn 512
Stream 513
SupportedQueryFeatures 513
SystemType 513
TypeName 514
User 514
UserCondition 517
UserField 517
UserSearch 518
VersioningOption 519
ViewParameter 520
Volume 523
Weekly 525

Part 3
Working with BIRT iHub integration APIs

Chapter 10
Using Java Report Server Security Extension 529
About the Java Report Server Security Extension 530
Implementing the Java RSSE interface 530
About installing a Java RSSE application 531
Installing a Java RSSE application 532
Configuring and deploying an LDAP configuration file 533
Installing the page-level security application 540
Migrating a Java RSSE application to a new Actuate release 540
Using page-level security 541
Creating an access control list (ACL) 542
Deploying a report to an Encyclopedia volume 543
SOAP-based Report Server Security Extension (RSSE) operations 545
Authenticate 546
DoesGroupExist 546
DoesRoleExist 547
DoesUserExist 547
GetConnectionProperties 548
GetTranslatedRoleNames 549
GetTranslatedUserNames 549
GetUserACL 550
GetUserProperties 550
GetUsersToNotify 551
PassThrough 551

SelectGroups 552
SelectRoles 553
SelectUsers 554
Start 555
Stop 557
SOAP-based Report Server Security Extension (RSSE) data types 557
Arrays of data types 557
Permission 558
PropertyValue 559
TranslatedRoleNames 559
User 560

Chapter 11
Using Actuate logging and monitoring APIs 563
About usage logging and error logging extensions 564
Installing and using usage and error logging extensions 564
Customizing the usage logging extension 567
Customizing the error logging extension 568
About the usage log 568
About types of recorded events 569
Understanding a usage log entry 569
About the error log 571
Understanding an error log entry 572
About BIRT iHub error messages 574
About BIRT iHub usage and error log consolidator 574

Chapter 12
Actuate logging and monitoring functions 591
About Usage Logging Extension functions 592
AcIsThreadSafe 592
AcLogUsage 592
AcStartUsageLog 592
AcStopUsageLog 593
About Error Logging Extension functions 593
AcIsThreadSafe 593
AcLogError 593
AcStartErrorLog 593
AcStopErrorLog 594
About the Performance Monitoring API 594
ArrayOfCounterInfo 594
CounterInfo 594
GetAllCounterValues 595

GetCounterValues 595
ResetCounters 596

Chapter 13
Aging and archiving Encyclopedia volume items 597
Automating report archival and removal 598
About Actuate Online Archive Driver 598
Configuring the Online Archive Driver 599
Understanding aging and archiving rules for items in an Encyclopedia volume 603
Understanding precedence in archiving 604
Aging and archiving an item using the Actuate Information Delivery API 604
Setting and updating autoarchive rules using IDAPI 604
Setting default autoarchive rules when creating a folder 606
Setting autoarchive rules when creating a job schedule 606
Updating autoarchive rules for a file or folder 607
Updating autoarchive rules for a job output file 608
Updating the autoarchive rules for a file type in a folder or volume 608
Setting an autoarchive schedule when updating an Encyclopedia volume 609
Starting an archive process for an Encyclopedia volume 610
Retrieving autoarchive rules for a file or folder 610
Setting job notice expiration for all users 611
Setting job notice expiration for a user 612

Chapter 14
Archiving APIs 613
SOAP-based archiving API operations 614
DeleteExpiredFiles 614
EndArchive 614
GetNextExpiredFiles 615
StartArchive 616
SOAP-based archiving data types 617
ArrayOfFileInfo 617
ArrayOfPermission 617
FileAccess 617
FileInfo 618
Permission 619

Chapter 15
Customizing installation on Windows systems 621
Extracting an Actuate product installation package 622
Modifying the installed files and registry entries 622
Localizing the installation 624

Creating a silent installation 628
Specifying version information 630
Specifying license information 631
Customizing installation dialog boxes 631
Specifying dialog box information 631
Encrypting dialog box information 632
Using acencrypt 632
Performing a silent installation 633
Performing a silent installation removal 635
Performing a silent removal of Actuate Localization and Online Documentation 636

Chapter 16
Customizing installation on UNIX and Linux systems 637
Extracting an Actuate product installation package 638
Creating a silent installation 638
Modifying the parameter template file 639
About isinstall.cfg 639
Modifying isinstall.cfg 639
Performing a silent installation 640
Performing a silent installation removal 641

Appendix A
Text string limits in Actuate operations 643

Index 647

About *Using BIRT iHub Integration Technology*

Using BIRT iHub Integration Technology provides information about server-side programming using the XML-based Actuate Information Delivery API. This guide includes an introduction to the concepts required to work with the API, covers Actuate's logging, auto archiving, and open server capabilities, and provides information about using the Java Report Server Security Extension (RSSE).

Using BIRT iHub Integration Technology includes the following chapters:

- *About Using BIRT iHub Integration Technology.* This chapter provides an overview of this guide.
- *Part 1. Introduction to the Actuate Information Delivery API.* This part describes how to work with Actuate Information Delivery API.
- *Chapter 1. Understanding the Information Delivery API and schema.* This chapter introduces the features of the API and describes the Actuate Web Services Description Language (WSDL) schema.
- *Chapter 2. Constructing a SOAP message.* This chapter discusses the elements of Actuate SOAP (simple object access protocol) messages.
- *Chapter 3. Understanding Actuate Information Delivery API operations.* This chapter summarizes the web services available through the API.
- *Chapter 4. Running, printing, and viewing a document.* This chapter discusses running a design, scheduling design generation, sending job notifications, selecting items for viewing, and other tasks the Factory and Viewing services support. chapter
- *Chapter 5. Administering an Encyclopedia volume.* This chapter discusses the functionality of the Encyclopedia service, including security and authentication, Encyclopedia- and volume-level administrative tasks, and searching within a volume.

- *Part 2. Developing Actuate Information Delivery API applications.* This part provides grouped lists of Actuate Information Delivery API operations that are frequently used together and complete descriptions of the Actuate Information Delivery API data types and operations.
- *Chapter 6. Developing Actuate Information Delivery API applications using Java.* This chapter describes how to use the Actuate Information Delivery API framework to create client applications that request Actuate iHub to perform administration, search, batch, and transaction operations, upload or download files, and schedule a custom event, using the Apache Axis development environment.
- *Chapter 7. Developing Actuate Information Delivery API applications using Microsoft .NET.* This chapter describes how to use the Actuate Information Delivery API framework to create client applications that request Actuate iHub to perform administration, search, batch, and iHubtransaction operations, and upload or download files, using the Microsoft .NET development environment.
- *Chapter 8. Actuate Information Delivery API operations.* This chapter provides an alphabetical listing of the Information Delivery API operations, including a general description, schema, and a description of each element.
- *Chapter 9. Actuate Information Delivery API data types.* This chapter contains an alphabetical listing of the Actuate Information Delivery API data types, including a general description of each data type, its schema, and a description of each element.
- *Part 3. Working with BIRT iHub integration APIs.* This part describes various integration technologies, such as error, usage, and performance logging, archiving Encyclopedia volume items, Actuate open server technology, and how to implement external security. This part also provides descriptions of the Actuate logging APIs, archiving functions and operations, and Java Report Server Security Extension (RSSE) functions and operations.
- *Chapter 10. Using Java Report Server Security Extension.* This chapter describes how to create and install an Actuate iHub Java Report Server Security Extension (RSSE) application as a web service. Using the Java RSSE framework, a developer can create an application that provides external authentication, external registration, and page-level security.
- *Chapter 11. Using Actuate logging and monitoring APIs.* This chapter discusses how to use the Actuate Error Logging, Usage Logging, and Performance Monitoring APIs.
- *Chapter 12. Actuate logging and monitoring functions.* This chapter provides an alphabetical listing of the functions and operations of the Error Logging, Usage Logging, and Performance Monitoring APIs, including a general description, syntax or schema, and a description of each parameter or element.

- *Chapter 13. Aging and archiving Encyclopedia volume items.* This chapter discusses aging rules and explains how to automate archiving and removal using the Actuate Information Delivery API.
- *Chapter 14. Archiving APIs.* This chapter provides an alphabetical listing of the SOAP-based archiving API operations, including a general description, syntax or schema, and a description of each parameter or element.
- *Chapter 15. Customizing installation on Windows systems.* This chapter discusses customizing Actuate product installations in a Windows environment and performing a silent installation.
- *Chapter 16. Customizing installation on UNIX and Linux systems.* This chapter describes customizing Actuate product installations in a UNIX and Linux environment and performing a silent installation.
- *Appendix A. Text string limits in Actuate operations.* This appendix lists the maximum field lengths for text elements in iHub Information and Management Consoles for elements the Actuate Information Delivery API creates.

Part One

Introduction to the Actuate Information Delivery API

Chapter

1

Understanding the Information Delivery API and schema

This chapter contains the following topics:

- About the Actuate Information Delivery API
- About web services and WSDL
- Understanding the elements of iHub's WSDL schema
- Accessing the Actuate schema using a web browser

About the Actuate Information Delivery API

The Actuate Information Delivery application programming interface (API) supports integrating and administering BIRT iHub using extensible markup language (XML) and the simple object access protocol (SOAP). Using the Information Delivery API, developers create applications that perform such tasks as uploading and downloading files, generating a document and scheduling document generation, sending an e-mail notification when a job completes, managing the users and security roles in an Encyclopedia volume, and working with external libraries.

SOAP is the underlying layer that provides a messaging framework for web services. A web service supplies functionality and capability over the Internet which assist in the creation of applications. Web services support integration of loosely coupled applications that are language neutral and platform independent, with application deployment using a standard transport protocol such as HTTP. Users, developers, and administrators can discover, describe, and invoke these applications in a distributed environment.

The Actuate Information Delivery API has the following features:

- Platform- and language-independent access to Actuate web services

  Using SOAP messaging, Actuate's web services integrate into applications developed in Java, Visual Basic, C++, C#, and other programming languages. The SOAP framework translates XML messages to the language of the calling application.

  Deployment of these applications can be across multiple platforms, including UNIX, Windows, or Linux, and integrate with web technologies such as J2EE and Microsoft Visual Studio .NET.

- Comprehensive Encyclopedia volume administration

  The Encyclopedia volume is the central repository for the design files, folders, and other items that BIRT iHub stores and manages. The Actuate Information Delivery API, can manage the items in an Encyclopedia volume from a single machine, and can send success or failure notices for immediate and scheduled jobs using simple mail transfer protocol (SMTP), or UNIX sendmail. The notifications can include an attachments or embedded files.

- Open infrastructure

  The Actuate Information Delivery API supports BIRT iHub's open server infrastructure. This infrastructure generates, distributes, and manages Actuate and third-party designs and integrates the designs into an Encyclopedia volume. The API also automates extracting parameters from a third-party executable file when integrating the file into an Encyclopedia volume.

- Localization support

  By setting the Locale parameter in the header of a SOAP request, localization generates data in a language other than the BIRT iHub default language. The data display uses the currency format, date format, and other conventions for specific locales.

About web services and WSDL

Actuate's web services support interactions with BIRT iHub, from running and viewing designs to managing Encyclopedia volume items. Actuate describes its services using Web Services Description Language (WSDL), an XML schema that provides the structure for requests to and responses from BIRT iHub.

A WSDL schema provides an abstract definition of the operations that a web service supports. The schema describes web services by providing such information as the name of the service, its transport protocol, data types, messages and operations, and input and output parameters. The schema binds these definitions to a concrete network protocol and message format. This schema resides on BIRT iHub.

iHub's WSDL schema serves as an interface to applications that integrate with BIRT iHub. The schema encompasses all web services accessible using the Actuate Information Delivery API. A developer can use the schema to generate a code library that contains the classes, including proxies, that the developer uses to write an application that communicates with a web service.

A developer can use an integrated development environment (IDE) to develop an application that uses the Information Delivery API schema. Actuate supplies WSDL files for the Apache Axis and Microsoft .NET development environments.

The elements of iHub's schema are case-sensitive. Capitalization must occur as the schema indicates. For example, to use the AuthId element of the complex data type Header, write AuthId instead of AuthID or authid.

Understanding the elements of iHub's WSDL schema

A WSDL file defines every element used in a SOAP message. Figure 1-1 shows the basic structure of a WSDL file. Additional details about how the Actuate Information Delivery API defines each element in the following sections.

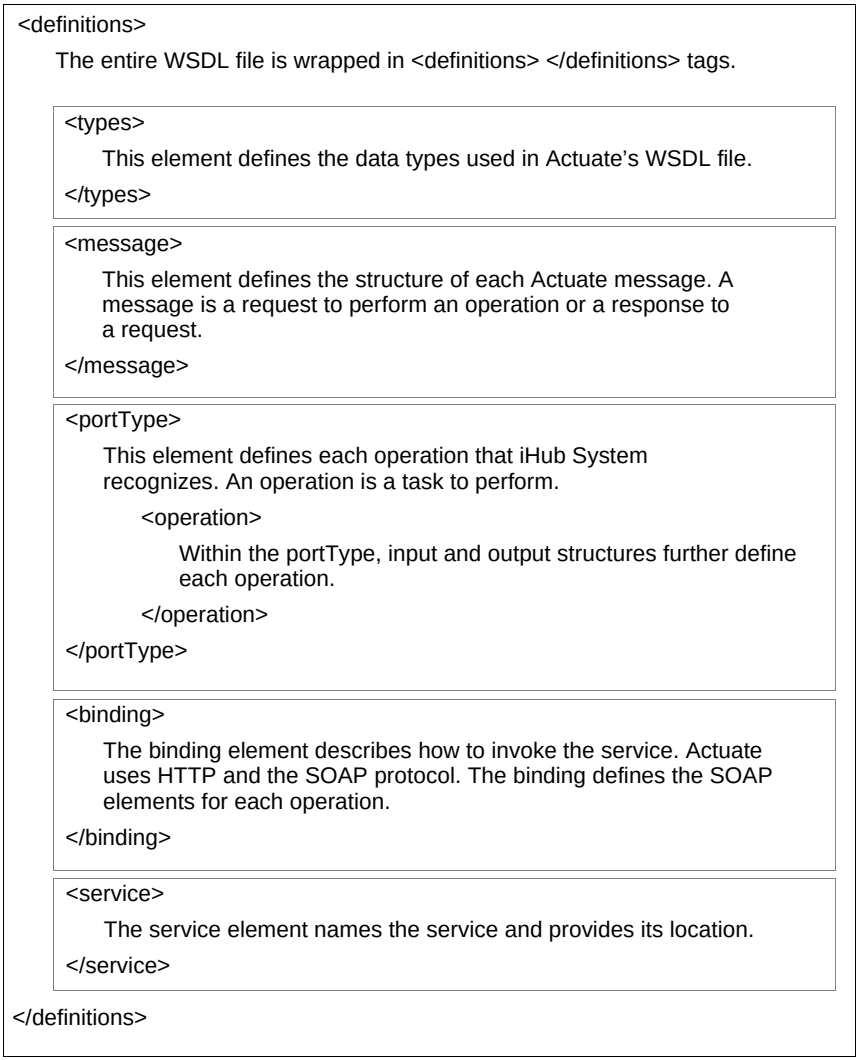


Figure 1-1 The basic structure of a WSDL file

About the definitions element

Because it is a definitions document, the WSDL schema begins with a definitions element. The opening tag defines the following Actuate namespace and attribute declarations:

```
<definitions
name="ActuateAPI"
xmlns="http://schemas.xmlsoap.org/wsdl/"
```

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:typens="http://schemas.actuate.com/actuate11"
xmlns:wsdlns="http://schemas.actuate.com/actuate11/wsdl"
targetNamespace="http://schemas.actuate.com/actuate11/wsdl">
```

Table 1-1 describes each declaration. These declarations are subject to change over time.

Table 1-1 Namespace and attribute declarations for the definitions element

Declaration	Description
name="ActuateAPI"	Names the Actuate service
xmlns="http://schemas.xmlsoap.org/wsdl/"	Defines a namespace for the WSDL specification to which Actuate adheres
xmlns:xsd="http://www.w3.org/2001 /XMLSchema"	Defines a namespace prefix, xsd, for the XML schema standard
xmlns:soap="http://schemas.xmlsoap.org/wsdl /soap/"	Defines a namespace prefix, soap, for the SOAP specification to which Actuate messages adhere
xmlns:typens="http://schemas.actuate.com /actuate11"	Defines a namespace prefix, typens, for the Actuate 11 XML schema
xmlns:wsdlns="http://schemas.actuate.com /actuate11/wsdl"	Defines a namespace prefix, wsdlns, for Actuate 11 web services
targetNamespace="http://schemas.actuate.com /actuate11/wsdl"	Scopes messages to the Actuate WSDL file for Actuate 11

About data type definitions

The types element of a schema describes every complex data type that Actuate web services recognize. The types element begins with the following declarations:

```
<types>
   <xsd:schema
      xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      targetNamespace="http://schemas.actuate.com/actuate11"
      elementFormDefault="qualified">
```

Two of these declarations are unique to the types element:

- xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/" specifies the encoding scheme for serializing and deserializing SOAP messages.
- elementFormDefault indicates whether the target namespace must qualify all locally declared elements in the instance document. A value of qualified

requires checking the target namespace to see that the instance document conforms to the target namespace element declarations and type definitions. A value of unqualified does not require checking.

The types element gives a data type a name and defines the structure. The types element describes whether there is a required sequence for the elements that define the structure. The following example shows the complete description of the complex data type for the Login request:

```
<xsd:complexType name="Login">
   <xsd:sequence>
      <xsd:element name="User" type="xsd:string"/>
      <xsd:element name="Password" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="EncryptedPwd" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Credentials" type="xsd:base64Binary"
         minOccurs="0"/>
      <xsd:element name="Domain" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="UserSetting" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="ValidateRoles" type="typens:ArrayOfString"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:element name="Login" type="typens:Login"/>
```

In the preceding example:

- The xsd: namespace prefix refers to the version of the XML schema that Actuate uses. Actuate reserves the xsd: namespace prefix to refer to the 2001 version of the standard XML schema.

  ```
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  ```

  The xsd: prefix appears in every tag in the schema.

- The complex data type is Login.

  Child elements define this data type. Each child element has attributes such as the data type, the name, and the minimum or maximum number of occurrences. If the child element defines a data value that the data type requires, there is no minOccurs attribute. In this example, User is the only required element.

- Because <sequence> </sequence> tags enclose this element list, you must use these elements in the sequence shown. If <all> </all> tags enclose the elements, they can appear in any order. If <choice> </choice> tags enclose the elements, you can choose one element.

About message definitions

The message element describes the parts of each request and response message in the Actuate Information Delivery API.

Actuate provides versions of its schema for the Microsoft .NET and Apache Axis environments. These environments require slightly different syntax for WSDL definitions. For example, a message definition contains a header element in the .NET version but not in the Apache Axis version. The code examples in this chapter use the Apache Axis version.

The following example shows the Apache Axis message description for a request to search for jobs:

```
<message name="SelectJobs">
   <part name="Request" element="typens:SelectJobs" />
</message>
```

Like most requests, this one has a corresponding response.

```
<message name="SelectJobsResponse">
   <part name="Response" element="typens:SelectJobsResponse" />
</message>
```

Operations that do not require a response do not have a corresponding response.

About the portType definition

The portType element defines a structure for each operation in the Actuate schema. A unique name identifies the portType:

```
<portType name="ActuateSoapPort">
```

For most operations, portType defines an input message, the request, and an output message, the response.

The following example shows the portType definition for ActuateSoapPort:

```
<operation name="GetFileACL">
   <input message="wsdlns:GetFileACL"/>
   <output message="wsdlns:GetFileACLResponse"/>
</operation>
```

If an operation does not require a response, portType defines only an input message. In the Microsoft .NET version of the schema, the operations also include a header element.

About the binding definition

The binding element defines how the operations and messages in the schema communicate. Actuate defines its binding using the following attributes:

- name="ActuateSoapBinding"

- type="wsdlns:ActuateSoapPort"
- style="document"
- transport="http://schemas.xmlsoap.org/soap/http"

The following example shows the binding definition for ActuateSoapBinding:

```
<binding name="ActuateSoapBinding" type="wsdlns:ActuateSoapPort">
<soap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http"/>
```

Within the <binding> </binding> tags, input and output child elements define the bindings for each operation that the portType section describes. Each input message consists of a SOAP header with attributes and a SOAP body with attributes. Each output message consists of a SOAP body with attributes.

The following example shows the input and output child elements for the Login operation:

```
<operation name="Login">
   <soap:operation soapAction="" />
      <input>
         <soap:body use="literal" parts="Request"/>
      </input>
      <output>
         <soap:body use="literal"/>
      </output>
</operation>
```

These elements specify that Actuate does not use soapAction, a required element for SOAP messages. They further show that Actuate operations are literal messages that do not use separate wrappers for each element.

About the service definition

The WSDL schema presents its various operations as a single web service. The service element defines that service, ActuateAPI. The service definition includes:

- The name of the requested service.

  ```
  service name="ActuateAPI"
  ```

- The name of the port through which the client accesses iHub's web services.

  ```
  port name="ActuateSoapPort"
  ```

- The binding definition expressed as a namespace.

  ```
  binding="wsdlns:ActuateSoapBinding"
  ```

- The location of the port, defined as host_name:port_number, expressed as a valid URL.

  ```
  soap:address location="http://localhost:8000"
  ```

The following example shows the service definition for ActuateAPI:

```
<service name="ActuateAPI">
   <port name="ActuateSoapPort"
      binding="wsdlns:ActuateSoapBinding">
      <soap:address location="http://localhost:8000"/>
   </port>
</service>
```

Accessing the Actuate schema using a web browser

To access the Actuate 11 WSDL file using a web browser, use the following URL:

`http://localhost:8000/wsdl/v11/axis/all`

where

- localhost is the local BIRT iHub.
- 8000 is the port to which the SOAP endpoint binds.
  Use port 8000 if you use only the current Actuate release of BIRT iHub. Use port 9000 if you use multiple Actuate versions.

The preceding URL displays the WSDL document, as shown in Figure 1-2.

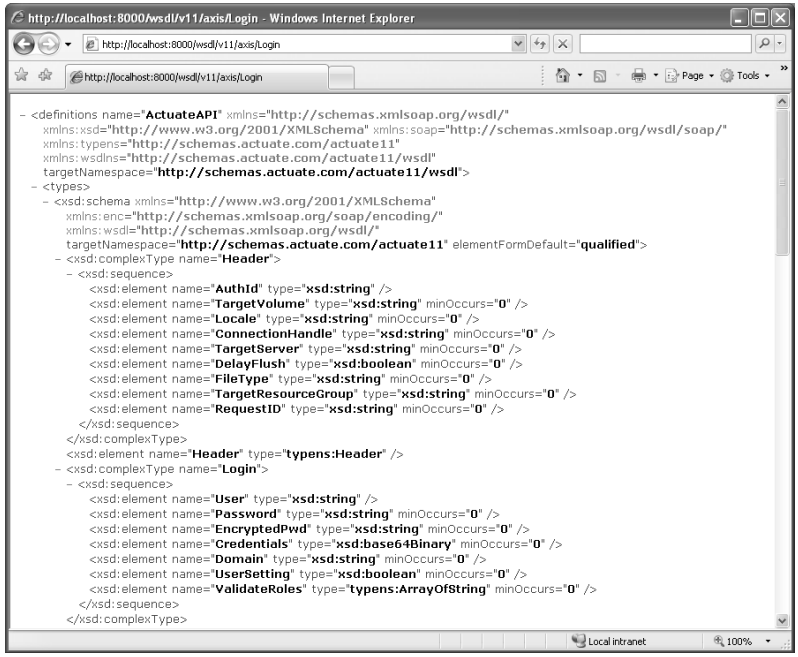


Figure 1-2 Login operation schema

Chapter

2

Constructing a SOAP message

This chapter consists of the following topics:

- About SOAP messaging
- Calling an Actuate web service
- About SOAP message elements
- About SOAP Fault messages

About SOAP messaging

This chapter describes the elements of Actuate simple object access protocol (SOAP) messages. The Actuate Information Delivery API uses SOAP messaging in the request and response pattern for communications between the client and BIRT iHub. SOAP messages are written in XML to ensure standard message formatting and standard data representation.

The principal advantage of SOAP is that it supports communications among applications written in different programming languages and running on different platforms. SOAP supports Java, Visual Basic, C++, C#, and other programming languages. It operates on Windows, UNIX, Linux, Mac, and other operating systems.

Certain messages in the Actuate Information Delivery API can be composite messages, supporting multiple operations in a single message.

The Actuate Information Delivery API packages an XML request into a SOAP envelope and sends it to the BIRT iHub using a hypertext transfer protocol (HTTP) connection. Although a SOAP message can use other transport mechanisms, Actuate supports HTTP because this protocol is ubiquitous and because it simplifies external firewall management.

The client application sends the request and reads the response in the client's native language. The system's SOAP endpoints, ports that accept SOAP messages, listen for requests and direct them to the appropriate BIRT iHub node.

As with any other XML document, a SOAP message must be well formed and valid. A well-formed message has a single root, is correctly nested, and displays tags in starting and ending pairs. Valid XML is well formed and adheres to a schema. XML instruction is outside the scope of this book.

Calling an Actuate web service

When accessing Actuate's web services, you create a library of proxy objects for the client application. In Java, a proxy object is a class implementation in a JAR file. In C#, a proxy is a CS file.

Use a proxy object directly. Actuate does not support subclassing an Actuate Information Delivery API class generated from an Actuate WSDL document.

Access proxy objects using a request and response pattern. As Figure 2-1 shows, the client uses a proxy object to send a SOAP request to BIRT iHub and receives a response in the client's native language.

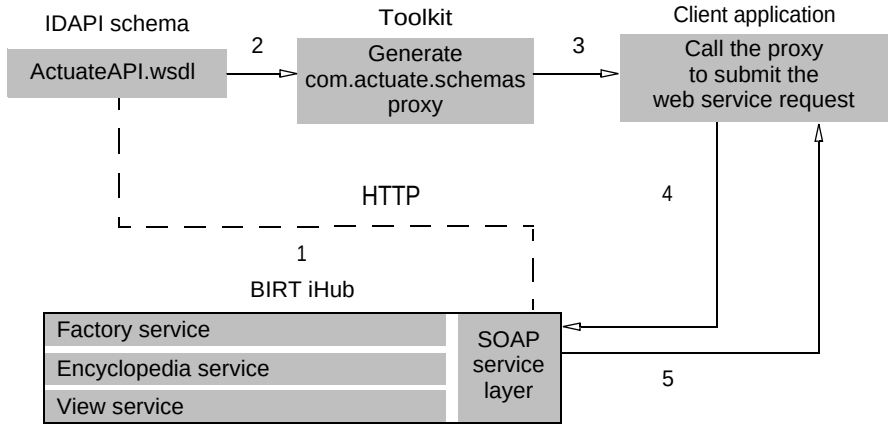


Figure 2-1 Calling an Actuate web service

In the sequence shown in Figure 2-1:

1 BIRT iHub sends the Information Delivery API schema over the web in response to a client query.

2 The client generates a proxy object that corresponds to a service or an operation in Actuate's schema.

At this point, you build or modify a client application.

3 The deployed client application calls a proxy object.

4 Using the proxy, the client generates a SOAP request, adds an HTTP header, and sends this serialized XML package to BIRT iHub over the web.

5 BIRT iHub processes the SOAP message header, deserializes the SOAP envelope, and invokes the appropriate service. In the preceding diagram, the Factory service processes the request.

6 The service serializes the result, creates the response XML, places the encoded result into a SOAP response, and returns the package to the client application. The application then extracts and decodes the result.

About SOAP message elements

The Actuate Information Delivery API uses the standard SOAP message structure. A message consists of an HTTP header followed by a SOAP envelope, header, and body. The SOAP envelope wraps the header and body elements. The HTTP header encloses the entire message, which goes over the web to a server that can accept SOAP messages.

The following sections provide details about each element of a SOAP message.

Understanding the HTTP header

This header is a mandatory element that specifies items such as the HTTP version, the host machine and port, the content type, and character set. The elements of an HTTP header can vary from message to message. The following example shows a typical HTTP header:

```
POST / HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related,
   text/*
User-Agent: Axis/1.4
Host: localhost:8080
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 1387
```

In the preceding example:

- POST routes the message to a servlet running on a web server, using HTTP version 1.0 or 1.1.

  The message determines which HTTP version to use. Version 1.0 treats an attachment as a single block of data. Version 1.1 supports sending chunked attachments.
- Content-Type specifies the message's media type. Set Content-Type to text/xml when calling an Actuate service.
- The default character set is UTF-8. To use the UTF-8 character set, it is not necessary to include this element in the HTTP header.
- Accept indicates the acceptable types of media in the response:
  - The application/soap+xml media type describes a SOAP message serialized as XML.
  - The application/dime media type supports processing a message either using MIME or by reference to a Uniform Resource Identifier (URI) that accesses a plug-in. A URI is a unique string that can be a Uniform Resource Locator (URL), a Uniform Resource Name (URN), or both. Using a URI ensures uniqueness.
  - The multipart/related media type indicates a compound object containing several inter-related parts in the body of a message.
  - The asterisk in text/* indicates the response can contain any type of text.
- User-Agent is the client that initiates the request.
- Host is the name of the host machine where the target BIRT iHub resides and, optionally, the port number.

- Cache-Control is a directive to any caching mechanism operating along the request-response transport layer. A no-cache directive keeps a cache from using the response to satisfy a subsequent request. This directive prevents the cache from returning a stale response to a client.
- Pragma is a directive to all recipients along the request-response transport layer. This no-cache directive requires the system to forward the original request to the target BIRT iHub even when a cached copy exists. Forwarding the original request prevents the transmission of a stale copy of a request.
- SOAPAction, a required element, tells the BIRT iHub that the message is a SOAP message. For an Actuate message, use a set of empty quotation marks for the SOAPAction value. The Encyclopedia volume determines the action based on the message body. The SOAPAction attribute requires empty quotes because it has no default value.
- Content-Length is the number of characters in the message.

Understanding the SOAP envelope

The SOAP envelope is a required element of each message. It defines an overall framework for the message and contains other elements of the message. The envelope contains namespace declarations that apply to the specific message that contains them and to any child elements of that message.

An XML namespace is a unique identifier for the elements and attributes of an XML document. When declaring an XML namespace in a SOAP envelope, define the rules by which the system interprets the content and structure of the message.

To ensure that a namespace is globally unique, the namespace must be a URI. A namespace does not have to point to a web site or online document.

In the following example, the namespace declarations indicate that the message adheres to specific XML and SOAP standards and identifies the version of the Actuate XML schema:

```
<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope
      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <soapenv:Header> … </soapenv:Header>
      <soapenv:Body>
         <GetVolumeProperties xmlns="http://schemas.actuate.com
            /actuate11">
            …
         </GetVolumeProperties>
      </soapenv:Body>
   </soapenv:Envelope>
```

In the preceding example:

- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap /envelope/" refers to the SOAP standard that the message elements follow. A SOAP envelope must have an element that references this namespace or a SOAP VersionMismatch error occurs.
- xmlns:xsd="http://www.w3.org/2001/XMLSchema" defines the scope of the XML namespace. In this case, the namespace indicates that the message is based on the World Wide Web Consortium XML schema initially published in 2001. This namespace is declared in every SOAP message.
- xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" refers to the specific instance of the XML schema. In order to use namespace attribute types in an XML document, first define the xsi namespace.
- xmlns="http://schemas.actuate.com/actuate11" in <soapenv:Body> refers to the Actuate XML schema version to use.

About XML namespace declarations

XML provides support for combining data from multiple sources. In the process, however, it is possible to create confusion to tag disparate message elements with the same name.

For example, if you use the <Target> tag for data about quarterly sales goals in a Sales and Marketing application, and you use the <Target> tag to denote fiscal year revenue targets in a Finance application, errors can occur. Combining data from these two sources into one XML document, can cause naming collisions, or duplications, which may result in error messages or erroneous data. To avoid these collisions, use XML namespace declarations.

When constructing a request to BIRT iHub, use the namespaces the Actuate XML schema specifies. To use a namespace, first declare it in a header and then refer to it using the appropriate tag in the message. The following example shows how to declare two new namespaces by placing them in the SOAP envelope header:

```
<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope
      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      xmlns:fin="http://www.actuate.com/soap/Finance"
      xmlns:sales="http://www.actuate.com/soap/Sales and
      Marketing">
```

These new namespaces define prefixes, fin and sales in the example above, as shorthand for data sources. Add the prefix to the appropriate tag name to indicate

which data source to use. Use a simple prefix, separated from the tag name by a colon, as shown in the following example:

```
<fin:Target> </fin:Target>
```

Understanding the SOAP header

The SOAP header contains authentication data, locale information, and other required or optional data. The SOAP header element is mandatory for calls to the BIRT iHub. Using the SOAP header, parser tools can locate key information without having to parse the entire message.

The following example shows a typical SOAP header with authentication and locale information:

```
<soapenv:Header>
   <AuthId>
      soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
      soapenv:mustUnderstand="0">
      u4yxAKHFJg9FY0JssYijJI5XvnpqDOPBOoWPbgRak20wIZIFDX6NY1o
      NsYg7RKzFt7GgtrOKqaas5HwLSkwhYEHEBl9PuZim4kDS5g==
   </AuthId>
   <Locale
      soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
      soapenv:mustUnderstand="0">en_US
   </Locale>
   <TargetVolume
      soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
      soapenv:mustUnderstand="0">
   </TargetVolume>
   <ConnectionHandle
      soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
      soapenv:mustUnderstand="0">
   </ConnectionHandle>
   <DelayFlush
      soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
      soapenv:mustUnderstand="0">true
   </DelayFlush>
</soapenv:Header>
```

The Actuate Information Delivery API extends the standard SOAP header to use the following elements.

AuthId

When the client logs in using the Actuate Information Delivery API, the system returns a system-generated, encrypted AuthId string in the Login response. All requests except Login requests must have a valid AuthId in the SOAP header. The header passes this ID to BIRT iHub for validation.

The example shows a typical AuthId. In subsequent requests in the same session, the AuthId identifier appears in the SOAP header. AuthId expires after a configurable period of time.

In the example, AuthId contains two attributes:

- soapenv:actor

  The value for actor, "http://schemas.xmlsoap.org/soap/actor/next", is a URI indicating that this message is for the first SOAP application capable of processing it.

- soapenv:mustUnderstand

  Indicates that the actor must understand and process the message and, if it can not, the actor must return a SOAP fault containing the value specified by the attribute. The mustUnderstand attribute can have a value of 0 or False, or 1 or True.

ConnectionHandle

An optional element that supports keeping a connection open to view a persistent document. ConnectionHandle is a session ID of the object.

ConnectionHandle supports phased downloading and viewing of a persistent report in the Encyclopedia volume to improve performance. When ConnectionHandle is present in the header, iHub System ignores the value for TargetVolume.

ConnectionHandle returns in two ways:

- As an element of a document generation response when the document is transient and progressive viewing is enabled. BIRT iHub System routes subsequent viewing requests to the BIRT iHub that generated the transient document.
- As a response to a viewing request, to ensure that subsequent requests by the same user go to the same View service until the report data changes. If the report name changes but the data remains the same, the View service displays the same report. If ConnectionHandle is present in the SOAP header, the system routes subsequent viewing requests to the same View service that returned the ConnectionHandle.

DelayFlush

A Boolean element that tells BIRT iHub to write updates to the disk when the value is False.

FileType

An element that specifies the file type to run, such as an HTML or BIRT design (.rptdesign) file. Specify FileType in the SOAP header for all execute, submit, and view IDAPI requests, such as ExecuteReport, SelectPage, or GetContent.

When running a design that specifies a TargetResourceGroup, specify FileType in the SOAP header. BIRT iHub System looks for a machine that can execute that file type. The default setting is for BIRT iHub System to look for any available iHub to manage a request, whether or not that iHub can run the requested file type.

Locale

BIRT iHub uses this element to format data using the language, date and time conventions, currency and other locale-specific conventions before returning the data to the client. If the client does not specify another locale, BIRT iHub System uses the client's default locale.

TargetResourceGroup

An optional element that supports assigning a synchronous report generation request to a specific resource group at run time. When using ExecuteReport to run a BIRT design that specifies a TargetResourceGroup, you must specify FileType in the SOAP header.

TargetServer

An optional element that refers to the BIRT iHub within a cluster to which to direct the request. Use TargetServer for requests pertaining to system administration, such as GetFactoryServiceJobs and GetFactoryServiceInfo.

TargetVolume

Refers to the Encyclopedia volume to which to direct the request. Use this element of the SOAP header to route a request to an Encyclopedia volume. In Release 10, TargetVolume is an optional element. In Release 11 and iHub, the Login message must specify the Encyclopedia volume name using TargetVolume in the SOAP header and Domain in the SOAP message body.

RequestID

An optional element that represents a unique value identifying the SOAP request.

Understanding the SOAP message body

The body of the message contains either the request for a specific operation, the response to a request, or an error message. The following example requests detailed data about users on the Encyclopedia volume:

```
<SOAP-ENV:Body>
   <SOAP-ACTU:SelectUsers
      xmlns:SOAP-ACTU="http://schemas.actuate.com/actuate11">
      <ResultDef SOAP-ENC:arrayType="xsd:String[5]">
         <String>Id</String>
         <String>Name</String>
         <String>EmailAddress</String>
```

```
        <String>Homefolder</String>
        <String>Description</String>
      </ResultDef>
      <Search>
        <CountLimit>201</CountLimit>
        <FetchSize>100</FetchSize>
        <FetchDirection>true</FetchDirection>
      </Search>
   </SOAP-ACTU:SelectUsers>
</SOAP-ENV:Body>
```

The response includes details for each attribute in the request, including the attribute name and data type, when the attribute takes effect, and whether it is required.

```
<SOAP-ENV:Body>
   <ACTU:SelectUsersResponse
      xmlns:ACTU="http://schemas.actuate.com/actuate11"
      xmlns="http://schemas.actuate.com/actuate11">
      <Users>
      <User>
        <Name>Administrator</Name>
        <Id>1</Id>
        <EmailAddress></EmailAddress>
        <HomeFolder></HomeFolder>
        <Description></Description>
      </User>
      <User>
        <Name>User0</Name>
        <Id>2</Id>
        <EmailAddress>User0@localhost</EmailAddress>
        <HomeFolder>/home/User0</HomeFolder>
        <Description></Description>
      </User>
      …
      </Users>
      <TotalCount>15</TotalCount>
   </ACTU:SelectUsersResponse>
</SOAP-ENV:Body>
```

About SOAP Fault messages

A SOAP Fault occurs when a request cannot be completed. Fault contains information identifying the source of the error or the component returning the error, the request, an error code, and a text description of the error.

In the following example, a request to download a file results in a SOAP Fault. The Description element contains a text error message and a reference to the requested file.

```
<SOAP-ENV:Body>
   <SOAP-ENV:Body>
      <SOAP-ENV:Fault
         xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
         envelope/"
         xmlns="http://schemas.xmlsoap.org/soap/envelope/">
         <faultcode>Server</faultcode>
         <faultstring>Soap Server error.</faultstring>
         <detail>
            <RequestName>DownloadFile</RequestName>
            <ErrorCode>3072</ErrorCode>
            <Description>
               <Message>Cannot find the specified file or folder,
                  or you do not have permission to access it.
               </Message>
            <Parameter1>/report/SampleReports.rptdesign</Parameter1>
            </Description>
         </detail>
      </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
```

Chapter

3

Understanding Actuate Information Delivery API operations

This chapter contains the topic About Actuate Information Delivery API operations.

About Actuate Information Delivery API operations

The Actuate Information Delivery API defines a syntax, in the form of an XML schema, for communicating with BIRT iHub using HTTP. The operations accessible using this API fall within the following functional areas:

- BIRT iHub
- Channels
- Encyclopedia volumes
- Files or folders
- Groups
- Information objects
- Jobs
- Searches
- Users, roles, and security

The following sections describe how to work with the Actuate Information Delivery API within these areas.

Working with BIRT iHub

BIRT iHub stores BIRT documents in an Encyclopedia volume, manages user information, handles report requests, and delivers BIRT documents.

Table 3-1 lists the operations that work with BIRT iHub.

Table 3-1 Operations for working with BIRT iHub

Operation	Description
CreateResourceGroup	Creates a resource group and sets its properties. A resource group specifies a set of Factory processes reserved to run only those jobs assigned to the group. Contains a required ResourceGroupSettings element to define properties of the resource group.
DeleteResourceGroup	Deletes a resource group other than the default resource groups. If a deleted group has a scheduled job assigned to it, the job remains in a pending state. If a job is running on an BIRT iHub assigned to a deleted resource group, the job completes.

Table 3-1 Operations for working with BIRT iHub (continued)

Operation	Description
GetAllCounterValues	Retrieves the values of all counters.
GetCounterValues	Retrieves information about specific counters.
GetFactoryServiceInfo	Retrieves general information about the Factory service, such as the name of the BIRT iHub on which it is running, the number of synchronous jobs currently queued on BIRT iHub, and the maximum number of synchronous jobs that can be in the queue.
GetFactoryServiceJobs	For synchronous jobs, returns a list of either the pending jobs or the jobs running on the Encyclopedia volume specified in TargetVolume in the SOAP header.
GetFormats	Retrieves a list of locales or a list of formats that BIRT iHub supports, such as PDF, DHTML, and XML.
GetResourceGroupInfo	Retrieves the resource group's settings. Returns an error if there is no matching resource group.
GetResourceGroupList	Returns two lists, one for synchronous resource groups, the other for asynchronous resource groups.
GetServerResourceGroup Configuration	Returns a list of resource groups on BIRT iHub and property settings for each group.
GetSystemMDSInfo	Retrieves the names and properties of a Message Distribution service (MDS) in a cluster or stand-alone BIRT iHub without authenticating the client. In a cluster, supports routing requests to an alternate MDS if the one to which the client connects fails. The request can indicate whether to retrieve MDS information only from BIRT iHubs that are currently online or from BIRT iHubs that are offline as well.
GetSystemPrinters	Retrieves a list of printers and printer details to which BIRT iHub connects.
GetSystemServerList	Retrieves a list of stand-alone and cluster BIRT iHubs, including the server name, state, and an error code and error description if the state is Failed.

(continues)

Table 3-1 Operations for working with BIRT iHub (continued)

Operation	Description
GetSystemVolumeNames	Retrieves the names of volumes in a stand-alone BIRT iHub or cluster system without authentication. Supports placing volume names on a Login page. The request can indicate whether to retrieve only the names of volumes currently online or all volumes in the cluster.
ODBOTunnel	Opens a connection to an OLAP server for ODBO API function.
Ping	A Ping request tests whether a specific component of BIRT iHub is operational and retrieves other diagnostic information about the component. The request must specify one of the following destinations to contact: ■ The Message Distribution service (MDS) ■ A BIRT iHub node running the Encyclopedia service (EE) ■ A BIRT iHub node running the Factory service (FS) ■ A BIRT iHub node running the View service (VS) ■ An Actuate open server driver (OSD) ■ A data source connection The request also can specify an action to perform on the destination, such as echoing payload data, reading a file, writing a temporary file, or connecting to a database. Not all actions are available for all destinations. Ping is available to an Encyclopedia volume administrator or a user in the Operator role.
ResetCounters	Resets the values of specific counters.
SetServerResourceGroup Configuration	Configures all the resource groups on an BIRT iHub.
UpdateResourceGroup	Updates resource group properties. Resource group name, type, or the name of the BIRT iHub on which the resource group runs are not updatable.

Working with channels

Channels receive electronic notifications and can display messages. Table 3-2 lists the operations that work with channels.

Table 3-2 Operations for working with channels

Operation	Description
CreateChannel	Creates a channel in an Encyclopedia volume. CreateChannel is accessed through the Administrate operation.
DeleteChannel	Deletes channels from the Encyclopedia volume. The request must specify whether to delete a single channel, a list of channels, or channels that match specific conditions. DeleteChannel is accessed through the Administrate operation.
SelectChannels	Retrieves a list of channels, channels that match specific conditions, or a single channel. The response returns channels to which the user making the request has read or write privileges.
UpdateChannel, UpdateChannelOperation, UpdateChannelOperation Group	Updates channel properties in the Encyclopedia volume. The request must specify the update activity to apply, such as updating general properties, and adding a user to one or more channels. The request must also specify whether the activity applies to a single channel, a channel list, or channels that match specific conditions. The response returns channels to which the user has read or write privileges. UpdateChannel is accessed through the Administrate operation.

Working with Encyclopedia volumes

The Encyclopedia volume is the main repository for applications and data. Operations exist to control data on the volume as well as the volume itself.

Encyclopedia volume administrators use Administrate to create, delete, update, copy, and move items within an Encyclopedia volume. An administrator manages the following items:

- Channels
- Files

- File types
- Folders
- Jobs, job notices, and job schedules
- Notification groups
- Security roles
- Users
- Volumes

The Administrate operation supports the ability to create composite operations that combine several transactions into one SOAP message. Grouping transactions reduces network traffic by streamlining the request and response process. This technique results in lower bandwidth use and increased throughput.

Update Administrate operations are also capable of acting as a composite. This feature supports the ability to implement multiple updates of Encyclopedia content in one SOAP message. A single SOAP message can contain multiple update operations, with each update operation having multiple updates within it. The following Administrate operations support multiple operations:

- UpdateChannel
- UpdateFile
- UpdateFileType
- UpdateGroup
- UpdateJobSchedule
- UpdateRole
- UpdateUser
- UpdateVolumeProperties

Typically, only an Encyclopedia volume administrator or a user in the Administrator role uses Administrate operations. The exception is that all users can change some of their own properties and all users can modify items they create.

Table 3-3 lists the operations that work with Encyclopedia volumes.

Table 3-3 Operations for working with Encyclopedia volumes

Operation	Description
Administrate	Packages operations that create, delete, copy, update, and move Encyclopedia volume items. Typically, only the Encyclopedia volume administrator or a user in the Administrator role uses Administrate operations.
ExecuteVolumeCommand	For a specified volume, begins a predefined command. The available commands are: ■ StartPartitionPhaseOut ■ StartArchive ■ SwitchToOnlineBackupMode ■ SwitchToNormalModes
GetVolumeProperties	Retrieves the properties of an Encyclopedia volume. GetVolumeProperties can also return schedule, printer options, autoarchive settings, archive library, and other properties.
Transaction	A packaging mechanism for Administrate operations. If a failure occurs anywhere in a transaction, all operations in the transaction fail.
UpdateVolumeProperties, UpdateVolumeProperties Operation, UpdateVolumeProperties OperationGroup	Updates the properties of a specific Encyclopedia volume. The request must specify the update activity to apply, such as updating general properties or printer settings. UpdateVolumeProperties is accessed through the Administrate operation.

Working with files and folders

Operations for files and folders include the uploading and downloading, copying and moving of files to and from the Encyclopedia volume. Support also exists for the retrieval of file or folder properties, including identifying information, dependencies and autoarchive polices. Table 3-4 lists the operations that work with files and folders.

Table 3-4 Operations for working with files and folders

Operation	Description
CopyFile	Copies a file or folder in the working directory to a specified target directory. The request must specify whether to copy a single file or folder, a file or folder list, or all files or folders that match specific conditions. CopyFile is accessed through the Administrate operation.
CreateFileType	Creates a new file type in BIRT iHub. CreateFileType is accessed through the Administrate operation.
CreateFolder	Creates a folder in an Encyclopedia volume. CreateFolder is accessed through the Administrate operation.
DeleteFile	Deletes files or folders from the Encyclopedia volume. The request must specify whether to delete a single file or folder, a list of files or folders, or files or folders that match specific conditions. DeleteFile is accessed through the Administrate operation.
DeleteFileType	Deletes file types. The request must specify whether to delete a single file type, a list of file types, or file types that match specific conditions. DeleteFileType is accessed through the Administrate operation.
DownloadFile	Downloads a persistent file. File content streams to the client as a MIME attachment or is embedded in the response.
DownloadTransientFile	Downloads a transient file. The request requires a FileId and can also indicate whether to decompose a compound document. File content can be attached or embedded in the response.
GetDataExtraction Formats	Retrieves a list of DataExtractionFormat objects for a specific file type.
GetFileDetails	Retrieves the properties of a file or folder, including the FileId, the file's ACL, and its autoarchive rules.

Table 3-4 Operations for working with files and folders (continued)

Operation	Description
GetFileTypeParameter Definitions	Retrieves the definitions of parameters for a specific file type on the target Encyclopedia volume.
GetFolderItems	Retrieves a list of files or folders in an Encyclopedia volume folder.
MoveFile	Moves a file or folder from the working directory to a specified target directory in the Encyclopedia volume. The request must specify whether to move a single file or folder, a file or folder list, or all files or folders that match specific conditions. MoveFile is accessed through the Administrate operation.
SaveTransientReport	Saves a transient report to a specified file.
UpdateFile, UpdateFileOperation, UpdateFileOperation Group	Updates file or folder properties in the Encyclopedia volume. The request must specify the update task to apply, such as updating general properties, adding or removing dependencies, or changing parameters. UpdateFile must also specify whether the task applies to a single file or folder, a file or folder list, or files or folders that match specific conditions. Write privilege is required on a file or folder to update its properties. Grant privilege is required to update privileges on a file or folder. UpdateFile is accessed through the Administrate operation.
UpdateFileType, UpdateFileTypeOperation, UpdateFileTypeOperation Group	Updates file type properties in the Encyclopedia volume. The request must specify the update task to apply, such as updating general properties, and adding file type parameters. UpdateFileType must also specify whether the task applies to a single file type, a list of file types, or file types that match specific conditions. UpdateFileType is accessed through the Administrate operation.

(continues)

Table 3-4 Operations for working with files and folders (continued)

Operation	Description
UploadFile	Uploads a file to an Encyclopedia volume. The client can specify a version name and can indicate whether to create a new version of the file. Content can upload to BIRT iHub as a MIME attachment or an object embedded in the request.

Working with groups

Groups, or notification groups, consist of sets of users. Notifications go to these users when a job completes, depending on settable conditions. Notification groups also indicate the success or failure of completed jobs. Table 3-5 lists the operations that work with groups.

Table 3-5 Operations for working with groups

Operation	Description
CreateGroup	Creates a notification group in an Encyclopedia volume. CreateGroup is accessed through the Administrate operation.
DeleteGroup	Deletes one or more notification groups. The request must specify whether to delete a single group, a list of groups, or groups that match specific conditions. DeleteGroup is accessed through the Administrate operation.
SelectGroups	Retrieves a list of groups that match specific conditions or a single group.
UpdateGroup, UpdateGroupOperation, UpdateGroupOperation Group	Updates notification group properties in the Encyclopedia volume. The request must specify the update task to apply, such as updating general properties, and adding a user to one or more groups. It also must specify whether the task applies to a single group, a group list, or groups that match specific conditions. UpdateGroup is accessed through the Administrate operation.

Working with information objects and databases

An information object is a file that contains a SQL query. iHub can cache information objects. Table 3-6 lists the operations that work with information objects and databases.

Table 3-6 Operations for working with information objects and databases

Operation	Description
CloseInfoObject	Closes the data source connection for an information object.
CreateDatabaseConnection	Creates a connection to an Actuate Caching service (ACS) database.
DataExtraction	Extracts data from a specified object.
DeleteDatabaseConnection	Deletes an ACS database connection objects.
GetInfoObject	Retrieves a description of an information object.
ExecuteQuery	Reads an information object, optionally saving the resulting output file in the Encyclopedia volume.
FetchInfoObjectData	Retrieves the data source for an information object, using the connection that OpenInfoObject establishes.
GetDatabaseConnection Definition	Retrieves information about an ACS database connection object.
GetDatabaseConnection Parameters	Retrieves the connection parameter definitions that the database requires.
GetDatabaseConnection Types	Retrieves the list of available DBMS platforms.
GetMetaData	Retrieves the metadata describing a result set schema.
OpenInfoObject	Establishes a connection to an information object. The request specifies whether to embed the information object in the response or return it in blocks.
UpdateDatabase Connection	Updates an ACS database connection.

Working with jobs and reports

The Information Delivery API includes operations associated with running, printing, and scheduling a report. The API also supports open server processes that generate or print open server reports.

Specifically, the API supports:

- Generating a report immediately, with or without enabling transient report and progressive viewing features
- Submitting a report generation or print job to BIRT iHub on a specified schedule
- Submitting a request to generate or print the output of a query
- Cancelling a report generation or printing job
- Retrieving details of a job, such as input and output parameters, report parameters, schedules, and printer settings
- Printing a document using printers that connect to BIRT iHub
- Creating a parameter values file
- Retrieving parameters associated with an executable file
- Notifying users and notification groups when a job completes
- Getting detailed information about a job notification

Table 3-7 lists the operations that work with jobs and reports.

Table 3-7 Operations for working with jobs and reports

Operation	Description
CancelJob	Stops generation of a scheduled or immediate job. Returns either the status of the cancellation or an error message. For synchronous reports, returns one of three states: ■ Failed, meaning the cancellation did not succeed ■ Succeeded, meaning the cancellation succeeded ■ InActive, meaning the job completed before BIRT iHub received the cancellation request
CancelReport	Cancel synchronous report execution. CancelReport returns either the status of the cancellation or an error message.
CreateParameterValuesFile	Creates a report parameter values (.rov) file in an Encyclopedia volume. CreateParameterValuesFile contains a required ParameterValueList element.
CreateQuery	Generates a data object value (.dov) file. CreateQuery contains a required Query element, which provides details about the available columns, parameter definitions, sorting and filtering criteria, and other properties of the query.

Table 3-7 Operations for working with jobs and reports (continued)

Operation	Description
DeleteJob	Deletes one or more jobs. The request must specify whether to delete a single job, a list of jobs, or jobs that match specific conditions. DeleteJob is accessed through the Administrate operation.
DeleteJobNotices	Deletes one or more job notices. The request must specify whether to delete a single job notice, a list of job notices, or job notices that match specific conditions. DeleteJobNotices is accessed through the Administrate operation.
ExecuteReport	Runs a synchronous report. The WaitTime element supports specifying the minimum time for BIRT iHub to wait before returning a response. For transient reports, the request can include an Attachment element to indicate that the executable file to use is attached to the request. ExecuteReportStatus returns the status of the request.
ExportParameter DefinitionsToFile	Converts parameter definitions to an attached file that the client application can use as a report's parameter values file. The definitions can return as an attachment or an embedded object.
ExtractParameter DefinitionsFromFile	Extracts parameter definitions from a parameter values file.
GetContent	Retrieves the contents of a report for a specific component ID or component name and value. If the component ID is 0, returns the entire report.
GetCustomFormat	Extracts the results of calling the AcReport::GetCustomFormat method. For example, if the method creates an Excel file, GetCustomFormat retrieves the Excel file from an BIRT iHub node.
GetDocumentConversion Options	Retrieves a list of DocumentConversionOptions.
GetDynamicData	Retrieves a report's dynamic data.
GetEmbeddedComponent	When BIRT iHub returns an embedded URL in the report page, the browser requests the component the URL identifies. The component can be static data, dynamic data, or a style sheet.

(continues)

Table 3-7 Operations for working with jobs and reports (continued)

Operation	Description
GetJavaReportEmbedded Component	Retrieves an embedded component such as an image or a graph in a Java report document.
GetJavaReportTOC	Retrieves the table of contents of a Java report document.
GetJobDetails	Retrieves properties of a job, such as job name and ID, schedule, printer settings, resource groups to which the job is assigned, and notification list. The GroupingEnabled element indicates whether an end user can group data in a report or information object. GetJobDetails retrieves report parameters from the parameter values file for the job. If a scheduled job's RunLatestVersion element isTrue, GetJobDetails returns the combined parameters from the parameter values file and the latest report executable file. If a user requests details about a job the user did not submit, BIRT iHub returns a security error.
GetNoticeJobDetails	Retrieves details about a job notification. The response always includes a JobAttributes element, which lists general job properties such as the job name, ID, state, owner, job type, duration, and resource groups to which the job is assigned. The response also can include details about the input and output files, notification information, and schedules. GetNoticeJobDetails includes the GroupingEnabled element to indicate whether the end user can group data in a report or information object.
GetPageCount	Retrieves the number of pages in a report.
GetPageNumber	Retrieves the page number of a bookmark in a report.
GetParameterPickList	Retrieves the parameters names from a pick list in a report
GetReportParameters	Retrieves the parameter values file of a designated report. The designated report can be an Actuate or third-party file. The client can request parameter values using a JobId, a file ID, or file name.
GetStaticData	Retrieve the report's static data
GetStyleSheet	Retrieve the report's style sheet
GetSyncJobInfo	Retrieves information about a specific synchronous job, including the status of the report, an error description if the status is Failed, and details about pending or running jobs.

Table 3-7 Operations for working with jobs and reports (continued)

Operation	Description
GetTOC	Retrieves the table of contents for a report. Returns the table of contents in XMLDisplay format.
GetUserPrinterOptions	Retrieves the user's printer settings on BIRT iHub.
PrintReport	Prints a generated report. PrintReport can specify users, channels, or notification groups to notify at completion of printing.
SelectJavaReportPage	Returns a report page formatted in the specified display format indicated by the Page or Component element.
SelectJobNotices	Retrieves a single job notice or a list of job notices.
SelectJobs	Retrieves a list of jobs, jobs that match specific conditions, or a single job.
SelectJobSchedules	Selects all scheduled jobs matching the specified criteria.
SelectPage	Retrieves a page or range of pages to view. Pages are identified by number, page range, component ID, or other criteria.
SubmitJob	Submits a job request to run a report or run and print it in one operation. Jobs run synchronously or asynchronously. Printing occurs in asynchronous mode only. The job submitter can specify users, channels, or notification groups to notify at completion of the job. SubmitJob supports e-mail notification of success and failure, attaching the output report to the e-mail message, and choosing the format for the attachment. The job submitter also can override user preferences for the attachment format. If the request generates an information object, SubmitJob reads the executable file, generates a temporary query, and executes the request using the temporary file. SubmitJob supports choosing a resource group to process the job.

(continues)

Table 3-7 Operations for working with jobs and reports (continued)

Operation	Description
UpdateJobSchedule, UpdateJobScheduleOperation, UpdateJobSchedule OperationGroup	Updates scheduled jobs in an Encyclopedia volume. The request must specify the update activity to apply, such as changing printer settings, adding or removing schedules to a job, and changing notifications. It must also specify whether the activity applies to a single scheduled job, a list of scheduled jobs, or scheduled jobs that match specific conditions. UpdateJobSchedule supports e-mail notification of success and failure, attaching the output report to the e-mail, and choosing a format for the report attachment. Viewer preferences for the attachment format are overridable. If the scheduled job is a data object value (.dov) file, UpdateJobSchedule updates the DOV file using the latest query. Using UpdateJobSchedule, an Encyclopedia volume administrator or a user in the Administrator role can add the job to a resource group or change the resource group to which the job is assigned. UpdateJobSchedule is accessed through the Administrate operation.
WaitForExecuteReport	WaitForExecuteReport overrides the WaitTime setting of an ExecuteReport operation. For example, when an ExecuteReport request has a WaitTime of 2 seconds and the response indicates that the status is Pending, the client can send WaitForExecuteReport to keep waiting for report generation beyond 2 seconds.

Working with searches

The Information Delivery API supports searching for users, notification groups, security roles, channels, files, folders, resource groups, parameters, jobs, and other items. The API can retrieve information about all volumes in a BIRT iHub and all printers to which that BIRT iHub connects.

Performing a search operation

To target the items on which to act, you must apply a search condition to the operation. The com.actuate.schemas library contains many special classes for setting a search condition and implementing a search for an item in an Encyclopedia volume. Table 3-8 lists the operations that work with searches.

Table 3-8 Operations for working with searches

Operation	Description
GetSavedSearch	Retrieves a saved search.
SaveSearch	Saves the results of a search.
SearchReport	Searches a report for specific criteria.
SelectFiles	Searches for a file or folder in an Encyclopedia volume. Returns the file or folder as an embedded object or an attachment.
SelectFileTypes	Retrieves a list of file types, file types that match specific conditions, or a single file type.

The Information Delivery API does not support constructing composite search messages. For example, it is not possible to use one message to search for a user, a notification group, and a security role.

To search for user information in an external security system, such as a Lightweight Directory Access Protocol (LDAP) server or Microsoft Active Directory, call Java Report Server Security Extension (RSSE) API functions.

Using special characters in searches

Many operations support acting on one or more items in an Encyclopedia volume. A search condition can use a filter to eliminate extraneous data.

Use special characters as wildcard operators to create a flexible search expression that filters results to locate data sharing a similar pattern. If a search expression contains one or more special characters, a backslash (\) must precede each special character.

The following special characters are valid operators in a search expression:

- Ampersand (&)
- Asterisk (*)
- Backslash (\)
- Close square bracket (])
- Comma (,)
- Equal sign (=)
- Exclamation point (!)
- Greater than sign (>)
- Hyphen (-)
- Less than sign (<)
- Number sign (#)
- Open square bracket ([)
- Pipe sign (|)
- Question mark (?)
- Single quotation mark (')

For example, to search for the user name user#, you must type the following search expression:

```
user\#
```

If you type user#, the search returns user names that begin with user and end with a number, such as user1, user2, and so on.

Working with users, roles, and security

User, role, and security functions support the ability to create, delete, or query user and role capabilities. Security and authentication operations manage user access to files, folders, channels, licenses, and other items in an Encyclopedia volume. These operations also authenticate and log in the user or Encyclopedia volume administrator.

iHub does not support using nested parent or child roles with pass-through security. When associating pass-through security with an Encyclopedia volume security role, the role must have only individual users as members. For example, if the security roles Supervisor and Manager are defined externally, the Supervisor security role cannot be a child of the Manager security role.

Table 3-9 lists the operations that work with users, roles, and security.

Table 3-9 Operations for working with users, roles, and security

Operation	Description
CallOpenSecurityLibrary	Calls the Report Server Security Extension (RSSE) API AcRSSEPassThrough function or PassThrough message, which calls the RSSE for general purposes.
CreateRole	Creates a security role in the Encyclopedia volume. CreateRole is accessed through the Administrate operation.
CreateUser	Creates a user in the Encyclopedia volume. CreateUser is accessed through the Administrate operation.
DeleteRole	Deletes one or more security roles. The request must specify whether to delete a single security role, a list of security roles, or security roles that match specific conditions. DeleteRole is accessed through the Administrate operation.

Table 3-9 Operations for working with users, roles, and security (continued)

Operation	Description
DeleteUser	Deletes one or more users. The request must specify whether to delete a single user, a user list, or users that match specific conditions. DeleteUser is accessed through the Administrate operation.
GetChannelACL	Retrieves the access control list (ACL) for a specified channel. Use either a name or ID to identify channels. FetchHandle supports retrieving a large list in the response
GetConnectionProperty Assignees	Retrieves the users and roles for a file.
GetFileACL	Retrieves the ACL for a file or folder identified by either name or ID. FetchHandle supports retrieving a large list in the response.
GetFileCreationACL	Retrieves the user's ACL templates, which are the privileges applied when the user creates a file or folder in an Encyclopedia volume. The client can specify the user either by name or ID.
GetUserLicenseOptions	Retrieves the license options for a specific user.
Login	Authenticates a user to BIRT iHub. Requires a user name. Can also accept a password or other credentials. Always returns an AuthId for use in subsequent requests in the same session. Always returns a list of the Actuate features the user can access. Can also return the user's viewing preferences, valid security roles, and other user information. If the user is an Encyclopedia volume administrator, returns a list of the user's administrative privileges.
SelectRoles	Retrieves a list of roles, roles that match specific conditions, or a single role.
SelectUsers	Retrieves a list of users, users that match specific conditions, or a single user.
SetConnectionProperties	Sets the connection properties for a file based on user or role.
SystemLogin	Authenticates the system administrator in BIRT iHub. Requires a system password. Returns an AuthId for use in subsequent requests in the same session.

(continues)

Table 3-9 Operations for working with users, roles, and security (continued)

Operation	Description
UpdateOpenSecurityCache	Flushes the Encyclopedia volume's open security data and retrieves new data from an external security source. Use UpdateOpenSecurityCache to retrieve new data before the existing data expires. UpdateOpenSecurityCache is accessed through the Administrate operation.
UpdateRole, UpdateRoleOperation, UpdateRoleOperationGroup	Updates security role properties in the Encyclopedia volume. The request must specify the update activity to apply, such as updating general properties, and adding a user to one or more roles. It must also specify whether the activity applies to a single security role, a role list, or roles that match specific conditions. UpdateRole is accessed through the Administrate operation.
UpdateUser, UpdateUserOperation, UpdateUserOperationGroup	Updates user properties in the Encyclopedia volume. The request must specify the update task to perform, such as updating general properties, and adding the user to a notification group. It must also specify whether the task applies to a single user, a user list, or users that match specific conditions. UpdateUser is accessed through the Administrate operation.

Chapter

4

Running, printing, and viewing a document

This chapter contains the following topics:

- Generating or printing a document
- Running a synchronous report
- Running or printing a job
- Working with a resource group
- Retrieving and viewing data
- Managing a large list
- Working with a large message
- Delivering a multilingual document

Generating or printing a document

The Factory service manages document generation and printing operations. Use these operations to run and print a document, schedule or cancel a job, and work with a parameter value file. Specifically, the Factory service supports:

- Running a report or information object
- Scheduling a report or information object as a job
- Cancelling report or job generation
- Creating a parameter values file
- Extracting parameter definitions from or exporting parameter definitions to a file
- Getting information about parameters, jobs, and job notices
- Creating, updating, and deleting a resource group

Actuate's open server functionality extends the Factory service to generate a third-party report executable file.

To focus attention on the relevant portions of operations, most of the examples in this chapter show only the SOAP body of a message.

Running a synchronous report

Use ExecuteReport to run a synchronous report. Using InputFileName or InputFileId, you can specify a BIRT report or an external executable file to run. To save the output, set SaveOutputFile to True. Then, use RequestedOutputFile to indicate the output file's name, destination, or autoarchive rules. If the report uses parameters, use ParameterValues to set name-value pairs for each parameter.

The following ExecuteReport request generates a persistent report with progressive viewing enabled. The output file is not shared. It uses the default autoarchive rules of its file type.

```
<SOAP-ENV:Body>
   <ExecuteReport>
      <JobName>Sampling_Data</JobName>
      <InputFileId>170</InputFileId>
      <SaveOutputFile>true</SaveOutputFile>
      <RequestedOutputFile>
         <Name>/SamplingDataReport.rptdocument</Name>
         <AccessType>Private</AccessType>
         <ArchiveRule>
            <FileType>rptdocument</FileType>
```

```
      </ArchiveRule>
    </RequestedOutputFile>
    <ParameterValues>
      <ParameterValue>
        <Name>Currency</Name>
        <Value>2008</Value>
      </ParameterValue>
      <ParameterValue>
        <Name>Date</Name>
        <Value>8/4/2012</Value>
      </ParameterValue>
    …
    </ParameterValues>
    <ProgressiveViewing>true</ProgressiveViewing>
  </ExecuteReport>
</SOAP-ENV:Body>
```

In this request:

- InputFileId identifies the file to run. The input file resides on BIRT iHub and generates the output file.
- SaveOutputFile indicates whether the report is persistent or temporary. If you save the output file, the report is persistent.
- RequestedOutputFile indicates a path to the output file because this is a persistent report.
- AccessType indicates whether the file is private or shared.
- ParameterValues lists the name and value of any parameters the input file uses.
- ProgressiveViewing enables or disables progressive viewing. Using progressive viewing, the first page of the output appears as soon as it generates. Without progressive viewing, the first page appears when the entire report completes.

An ExecuteReport request returns the status of the report, an ObjectId that BIRT iHub generates, and the output file type. For a persistent report, ObjectId is valid until the user deletes the report. For a transient report, ObjectId is a temporary identifier that lasts for a configurable period of time.

An ExecuteReport response also returns a ConnectionHandle for a persistent report. The ConnectionHandle remains valid throughout the session.

```
<SOAP-ENV:Body>
  <ExecuteReportResponse>
    <Status>FirstPage</Status>
    <OutputFileType>RPTDOCUMENT</OutputFileType>
    <ObjectId>9015</ObjectId>
```

```
      <ConnectionHandle>g7whmBpUho+tg5MUYUgZxqVGrbtKH
      </ConnectionHandle>
   </ExecuteReportResponse>
</SOAP-ENV:Body>
```

Setting a time frame for an ExecuteReport response

Use ExecuteReport with WaitTime to set a time frame for a response to a report generation request. WaitTime requests a response from BIRT iHub within a specific period of time, even if report generation has not started or is incomplete. WaitTime specifies the minimum time BIRT iHub waits before it returns a response.

To avoid performance issues associated with frequent server time-outs, make WaitTime long enough for a typical report to generate. The ExecuteReport response is Pending if WaitTime is less than the time required to generate the first page of a progressive report or the time required to complete a non-progressive report.

The following example sets the wait time to 1 second and ProgressiveViewing to False. Using these settings, BIRT iHub returns a status message to the client if the entire report does not generate in 1 second.

```
<SOAP-ENV:Body>
   <ExecuteReport>
      <JobName>Detailed_Data</JobName>
      <InputFileName>/detail.rptdesign</InputFileName>
      <SaveOutputFile>true</SaveOutputFile>
      <RequestedOutputFile>
         <Name>detail.rptdocument</Name>
         <AccessType>Shared</AccessType>
      </RequestedOutputFile>
      <IsBundled>false</IsBundled>
      <ProgressiveViewing>false</ProgressiveViewing>
      <WaitTime>1</WaitTime>
   </ExecuteReport>
</SOAP-ENV:Body>
```

The response includes the status of the report generation request, the ObjectId, and the output file type.

```
<SOAP-ENV:Body>
   <ExecuteReportResponse>
      <Status>Pending</Status>
      <OutputFileType>RPTDOCUMENT</OutputFileType>
      <ObjectId>435</ObjectId>
   </ExecuteReportResponse>
</SOAP-ENV:Body>
```

An ExecuteReport request with WaitTime set returns one of the following report request status messages:

- Done means that report generation succeeded.
- Pending means that the report is either in the queue or in the process of generating.
- Failed means that the request to cancel did not succeed because of authorization errors or another reason.
- FirstPage means the first page of a progressive report is complete and the report is continuing to generate.

Waiting for report generation

Use WaitForExecuteReport to continue waiting for the report to generate after sending an ExecuteReport request and receiving a Pending status. For example, when an ExecuteReport request has a WaitTime of 2 seconds and the response indicates that the report status is Pending, a client can send WaitForExecuteReport to keep waiting for report generation beyond the specified wait time of 2 seconds.

WaitForExecuteReport waits until either the first page generates or the report is complete, depending on whether the user enables progressive viewing. If the report uses progressive viewing, the user can cancel after the first page generates. Otherwise, the user cannot cancel until the entire report completes. To avoid performance issues, the Factory service deletes the report from the queue if it takes too long to generate.

The WaitForExecuteReport request uses the ConnectionHandle and ObjectId from the ExecuteReport response.

```
<SOAP-ENV:Header>
   <AuthId>9FY0JssYijJI5XvkJqDOPBOoWPbgRak20wIZIFDUa</AuthId>
   <Locale>en_us</Locale>
   <ConnectionHandle>RYEMWxKREsxq0mQCiXCZHB8NiLDFwq1Q=
   </ConnectionHandle>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <WaitForExecuteReport>
      <ObjectId>435</ObjectId>
   </WaitForExecuteReport>
</SOAP-ENV:Body>
```

The WaitForExecuteReport response returns the ObjectId of the requested report, the status of the request, and the output file type.

```
<SOAP-ENV:Header>
   <ConnectionHandle>RYEMWxKREsxq0mQCiXCZHB8NiLDFwq1Q=
   </ConnectionHandle>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <WaitForExecuteReportResponse>
      <Status>Done</Status>
      <OutputFileType>RPTDOCUMENT</OutputFileType>
      <ObjectId>435</ObjectId>
   </WaitForExecuteReportResponse>
</SOAP-ENV:Body>
```

If progressive viewing is enabled, the response returns a status of FirstPage and the wait period ends.

Running a synchronous report that uses parameters

To run a report that uses parameters, use ExecuteReport and set the following properties of ParameterValues:

- Group is the group section in the report.
- Name is the parameter name.
- Value is the value to search for.

The following example shows the ExecuteReport request for a report that uses two parameters, customers_creditrank and offices_city. It asks for customers with a credit rank of C whose offices are in New York City.

```
<SOAP-ENV:Body>
   <ExecuteReport>
      <JobName>CREDIT_JOB</JobName>
      <InputFileId>369</InputFileId>
      <SaveOutputFile>true</SaveOutputFile>
      <RequestedOutputFile>
         <Name>/detail.rptdocument</Name>
         <ArchiveRule>
            <FileType>rptdocument</FileType>
         </ArchiveRule>
      </RequestedOutputFile>
      <ParameterValues>
         <ParameterValue>
            <Group>Customer Parameters</Group>
            <Name>DataSource::customers_creditrank</Name>
            <Value>C</Value>
         </ParameterValue>
```

```
        <ParameterValue>
           <Group>Office Parameters</Group>
           <Name>DataSource::offices_city</Name>
           <Value>NYC</Value>
        </ParameterValue>
     </ParameterValues>
     <ProgressiveViewing>true</ProgressiveViewing>
     <WaitTime>20</WaitTime>
   </ExecuteReport>
</SOAP-ENV:Body>
```

The response contains the same elements as a report that does not use parameters:

```
<SOAP-ENV:Body>
   <ExecuteReportResponse>
     <Status>FirstPage</Status>
     <OutputFileType>RPTDOCUMENT</OutputFileType>
     <ObjectId>5172</ObjectId>
     <ConnectionHandle>g7whmBpUho+tgppw=</ConnectionHandle>
   </ExecuteReportResponse>
</SOAP-ENV:Body>
```

Retrieving report parameter values

Use GetReportParameters to retrieve the parameter values of a specific report object value (.rov) file. Use one of the following identifiers to specify which ROV to use:

- JobId. BIRT iHub finds the associated ROV and reads the parameters from that file.
- The name or identifier of the file for which you want to retrieve parameter values. This file can be a BIRT or external report executable file, a parameter values file, or a third-party compound storage file.

For a date parameter, BIRT iHub returns parameter values in the General Date format, regardless of the DateTime settings in localemap.xml. For example, the General Date format is mm/dd/yyyy hh:mm:ss for the US English locale.

The following example requests parameters for a job:

```
<SOAP-ENV:Body>
   <GetReportParameters>
     <JobId>16</JobId>
   </GetReportParameters>
</SOAP-ENV:Body>
```

The response includes all parameters stored in the ROV:

```
<SOAP-ENV:Body>
   <GetReportParametersResponse>
      <ParameterList>
         <ParameterDefinition>
            <Group>Customer Parameters</Group>
            <Name>customers_creditrank</Name>
            <DefaultValue></DefaultValue>
            <IsRequired>false</IsRequired>
            <IsPassword>false</IsPassword>
            <IsHidden>false</IsHidden>
            <DisplayName>Credit Rank</DisplayName>
            <IsAdHoc>false</IsAdHoc>
         </ParameterDefinition>
         <ParameterDefinition>
            <Group>Customer Parameters</Group>
            <Name>customers_customName</Name>
            <DataType>String</DataType>
            <DefaultValue></DefaultValue>
            <IsRequired>false</IsRequired>
            <IsPassword>false</IsPassword>
            <IsHidden>false</IsHidden>
            <DisplayName>Customer Name</DisplayName>
            <IsAdHoc>true</IsAdHoc>
            <ColumnName>Name</ColumnName>
            <ColumnType>String</ColumnType>
         </ParameterDefinition>
         …
      </ParameterList>
   </GetReportParametersResponse>
</SOAP-ENV:Body>
```

If IsAdHoc is True for any parameter, the response returns ColumnName and ColumnType in the ParameterDefinition element of that parameter.

Running or printing a job

A job is a document generation or print operation that runs asynchronously. A job can run at a scheduled time or you can send a job to the queue immediately. Two operations support working with jobs and job schedules:

- Use SubmitJob to:
  - Create and schedule a job.
  - Set the job priority.

- Specify the requested output file type.
- Set the users, groups, or channels to notify.
- Determine the parameters to use when running the job.
- Provide printer options.
- Determine whether to retry a failed job.

- Use UpdateJobSchedule to:
  - Modify a schedule.
  - Change the type of operation and other job attributes.
  - Modify input and output parameters.
  - Add or delete notification recipients.
  - Change the number of versions to retain in BIRT iHub.
  - Modify search conditions.

Understanding SubmitJob

UseSubmitJob to specify a file to run or print. The input file can be BIRT design (.rptdesign) file, an external executable file, a report document such as a BIRT document (.rptdocument) file, or a report object value (.rov) file. If the input file is dependent on an executable file, BIRT iHub uses the executable file as the input.

The following example schedules a job, Sample Report, to run a report, SampleReport.rptdesign, and output a file, SampleReport.rptdocument:

```
<SOAP-ENV:Body>
   <SubmitJob>
      <JobName>SampleReport</JobName>
      <Priority>1000</Priority>
      <InputFileName>/report/SampleReport.rptdesign;1
      </InputFileName>
      <RunLatestVersion>false</RunLatestVersion>
      <RequestedOutputFile>
         <Name>/report/SampleReport.rptdocument</Name>
      </RequestedOutputFile>
      <Operation>RunReport</Operation>
      <ParameterValues
         …
      </ParameterValues>
      …
   </SubmitJob>
</SOAP-ENV:Body>
```

Specifying parameters for a job

To specify a version of a document to run or print, use SubmitJob and identify the version number, separating it from the file name with a semicolon (;). Use the version number, not the optional version name:

```
<InputFileName>Forecast.rptdesign;12</InputFileName>
```

Because SubmitJob supports only persistent reports, you must specify the output file name in the request, including the file type:

```
<RequestedOutputFile>
   <Name>Forecast.rptdocument</Name>
</RequestedOutputFile>
```

To indicate whether to run a report or run and print the report, set the Operation element to either RunReport or RunAndPrintReport:

```
<Operation>RunReport</Operation>
```

To run the job using the most recent version of a report, set RunLatestVersion to True and identify the input file using InputFileName. BIRT iHub ignores RunLatestVersion if you use InputFileId.

For a print job, you can specify settings for the printer using PrinterOptions. These printer settings include such details as the printer name, page size and orientation, scaling, number of copies to print, and pages to print.

About hidden, required parameters

When you run a report that uses a hidden, required parameter, BIRT iHub returns an error message if you do not provide the hidden parameter. The error message does not identify the hidden parameter. To determine whether a report uses hidden parameters, run GetReportParameters before submitting the job.

Scheduling report generation or printing

Use SubmitJob to run or print a job on a schedule. You can schedule any executable file to run or print immediately, daily, weekly, monthly, or on specific dates.

When a scheduled job cannot run because BIRT iHub is down, the job runs when BIRT iHub restarts. If a job has multiple pending occurrences when BIRT iHub restarts, only one instance runs.

The following example schedules a job to run once a week, using the highest priority, starting at midnight Pacific time every Monday from December 1, 2008, to December 1, 2009:

```
<SOAP-ENV:Body>
   <SubmitJob>
      <JobName>ForecastSchedule</JobName>
```

```
      <Headline>Quarterly forecast updates</Headline>
      <Priority>1000</Priority>
      <InputFileName>Forecast.rptdesign</InputFileName>
      <RunLatestVersion>true</RunLatestVersion>
      <RequestedOutputFile>
         <Name>Forecast.rptdocument</Name>
         <AccessType>Private</AccessType>
      </RequestedOutputFile>
      <Operation>RunReport</Operation>
      <Schedules>
         <TimeZoneName>PST</TimeZoneName>
         <ScheduleDetails>
            <JobScheduleDetail>
               <ScheduleType>Weekly</ScheduleType>
               <ScheduleStartDate>2008-12-1</ScheduleStartDate>
               <ScheduleEndDate>2009-12-1</ScheduleEndDate>
                  <Weekly>
                  <FrequencyInWeeks>1</FrequencyInWeeks>
                  <RunOn>Mon</RunOn>
                  <OnceADay>14:00:00</OnceADay>
               </Weekly>
            </JobScheduleDetail>
         </ScheduleDetails>
      </Schedules>
      <NotifyGroupsByName>
         <String>Sales Managers</String>
      </NotifyGroupsByName>
   </SubmitJob>
</SOAP-ENV:Body>
```

In this example:

- JobName is a title for the schedule.
- Headline is a title that a channel subscriber sees.
- Priority sets a priority for the job ranging from 0 to 1000, where 1000 is the highest priority.
- InputFileName identifies the executable file to use as input.
- RequestedOutputFile provides a file name and extension for the output. It also identifies the access type of the file, either Private or Shared.
- Operation identifies the task to schedule, either RunReport or RunAndPrintReport.
- NotifyGroupsByName sets the notification groups to notify of job success or failure. RunLatestVersion specifies whether to run or print the most recent version of the executable file. RunLatestVersion ignores any version numbers

in InputFileName.

- The JobScheduleDetail element of ScheduleDetails specifies the frequency of the run, the start and end dates for the schedule, the day and time of the run, and other details.

When the job succeeds, the SubmitJob request returns the JobId:

```
<SOAP-ENV:Body>
   <SubmitJobResponse>
      <JobId>145</JobId>
   </SubmitJobResponse>
</SOAP-ENV:Body>
```

For a successful job, users in the specified group receive a notification unless they choose not to do so. A report user can indicate whether BIRT iHub notifies him only of successful jobs, only of failed jobs, or of both successful and failed jobs.

When a job fails, BIRT iHub returns an error message and notifies those users who choose to receive a notification of job failure.

Working with a job notification

When an asynchronous job completes, BIRT iHub can notify a user, notification group, personal channel, and subscribed channel.

Notification tasks are suboperations of the SubmitJob and UpdateJobSchedule operations. To set a notification for a job, use SubmitJob and indicate the user, notification group, or channel to notify. To modify an existing notification, use UpdateJobSchedule.

Administrators and others who submit or update a job can indicate whether the job sends a notification. A report user can indicate whether BIRT iHub notifies him only of successful jobs, only of failed jobs, or of both successful and failed jobs.

The job submitter also can choose from a list of possible formats for the job output when executing a BIRT design (.rptdesign) or document (.rptdocument). This preference is not available for third-party reports.

About e-mail attachments

A user or notification group receives an e-mail notification when a job succeeds or fails. When the job completes successfully, the user or notification group can receive the output as an e-mail attachment or can link to the output. The person submitting the job can override user preferences for the output format. All users in a notification group receive the same output format.

To create an attachment, the Factory service runs an executable file to generate a report document. Then, the View service renders the document into the specified

attachment format. You can attach a document generated from any report executable file, including a third-party file type.

If you send a document as an attachment and the output document requires secure read privileges, BIRT iHub provides a link to the file instead of the attachment. A user who chooses the link must have secure read privileges to view the output. If the user does not have read privileges to the report, only the location of the document appears in the e-mail.

If a report user requests an e-mail attachment, he or she can choose either standard e-mail format or HTML format.

About notifying a channel

A channel receives an electronic notification and displays a message. If the channel is a personal channel, the message is visible only to the owner of the channel. If the channel is available to multiple subscribers, only those subscribers see the message. In either case, choosing a link in the message displays the output of a successful job.

Sending an e-mail notification using SubmitJob

To specify a user or notification group to receive a job notification, you can set one or more of the following elements of SubmitJob:

- NotifyUsersByName
- NotifyUsersById
- NotifyGroupsByName
- NotifyGroupsById

When you list the users or groups to notify, BIRT iHub determines their e-mail addresses based on the user names or IDs.

The following example represents portions of a SubmitJob operation using the NotifyUsersByName element:

```
<SOAP-ENV:Body>
   <SubmitJob>
      <JobName>forecast</JobName>
      …
      <Schedules>
         <TimeZoneName>PST</TimeZoneName>
         <ScheduleDetails>
            <JobScheduleDetail>
            …
            </JobScheduleDetail>
         </ScheduleDetails>
      </Schedules>
```

```
      <NotifyUsersByName>
         <String>Craig Osborne</String>
         <String>Ying Chen</String>
      </NotifyUsersByName>
      …
   </SubmitJob>
</SOAP-ENV:Body>
```

Sending an e-mail notification using UpdateJobSchedule

To modify a job's notification settings, set one or more of the following elements of UpdateJobSchedule:

- SetUserNotificationByName
- SetUserNotificationById
- SetGroupNotificationByName
- SetGroupNotificationById

The following UpdateJobSchedule operation uses SetUserNotificationByName and SetGroupNotificationByName to add three users and a group to the notification list. This operation is a transaction, which means that all updates must succeed for the operation to succeed.

```
<SOAP-ENV:Body>
   <Administrate>
      <Transaction>
         <TransactionOperation>
            <UpdateJobSchedule>
               <UpdateJobScheduleOperationGroup>
                  <UpdateJobScheduleOperation>
                     <SetAttributes>
                        <RunLatestVersion>false</RunLatestVersion>
                        <InputFileName>
                           /Regional Forecasts 2008
                           /forecast.rptdesign
                        </InputFileName>
                     </SetAttributes>
                     <SetParameters>
                        <OverrideRecipientPref>true
                        </OverrideRecipientPref>
                        <AttachReportInEmail>true
                        </AttachReportInEmail>
                        <SendEmailForSuccess>true
                        </SendEmailForSuccess>
```

```
                    <SendEmailForFailure>true
                    </SendEmailForFailure>
                    <EmailFormat>PDF</EmailFormat>
                    <SendSuccessNotice>true
                    </SendSuccessNotice>
                    <SendFailureNotice>true
                    </SendFailureNotice>
                 </SetParameters>
                 <SetUserNotificationByName>
                    <String>Claude Lacroix</String>
                    <String>Heinrich Richter</String>
                    <String>Ying Leung</String>
                 </SetUserNotificationByName>
                 <SetGroupNotificationByName>
                    <String>Sales Managers</String>
                 </SetGroupNotificationByName>
              </UpdateJobScheduleOperation>
           </UpdateJobScheduleOperationGroup>
           <Id>1</Id>
        </UpdateJobSchedule>
     </TransactionOperation>
  </Transaction>
</Administrate>
</SOAP-ENV:Body>
```

In this example:

- SendFailureNotice and SendSuccessNotice are True, so BIRT iHub sends e-mail for success and for failure.
- OverrideRecipientPref is True, so BIRT iHub overrides recipient preferences about whether to receive the attachment.
- EmailFormat requests the output in PDF format. You can request output in Excel, PDF, PostScript, Word, or PowerPoint format.

As with many administration operations, this request returns an empty response.

Notifying a channel using SubmitJob

To notify a channel using SubmitJob, set one of the following elements:

- NotifyChannelsByName
- NotifyChannelsById

For example, to notify the Sales Updates channel by name, include the following code in the request:

```
<SOAP-ENV:Body>
   <SubmitJob>
   …
      <NotifyChannelsByName>
         <String>Sales Updates</String>
      </NotifyChannelsByName>
   …
</SOAP-ENV:Body>
```

Notifying a channel using UpdateJobSchedule

To notify a channel using UpdateJobSchedule, set one of the following elements:

- SetChannelNotificationByName
- SetChannelNotificationById

The following operation uses SetChannelNotificationByName to specify two channels to notify, Accounts and Daily messages:

```
<SOAP-ENV:Body>
   <Administrate>
      <UpdateJobSchedule>
         <UpdateJobScheduleOperationGroup>
            <UpdateJobScheduleOperation>
               <SetAttributes>
                  <RunLatestVersion>false</RunLatestVersion>
                  <InputFileName>/office-info.rptdesign;
                  </InputFileName>
               </SetAttributes>
               <SetParameters>
               <RetryOption>VolumeDefault</RetryOption>
               </SetParameters>
               <SetChannelNotificationByName>
                  <String>Accounts</String>
                  <String>Daily messages</String>
               </SetChannelNotificationByName>
            </UpdateJobScheduleOperation>
         </UpdateJobScheduleOperationGroup>
         <Id>8</Id>
      </UpdateJobSchedule>
   </Administrate>
</SOAP-ENV:Body>
```

As with many administrative operations, this request returns an empty response.

Customizing an e-mail notification

Actuate provides a customizable e-mail notification template, acnotification.xml, which installs in the \iHub\etc directory. This file includes templates for success and failure messages. The template for success notifications includes a standard subject line for the e-mail, a simple message that forms the body of the e-mail, and a completion time stamp. It also provides a link to the output. The template for a failure message includes a variable to explain the reason for failure. You can customize the template by modifying the content portions of acnotification.xml.

Using the e-mail template for multiple locales

In an e-mail notification, the attachment returns in the language and with the locale conventions used to submit the request.

Because the default e-mail notification template uses UTF-8 encoding, you can also send the content of the subject line and the content of the body in multiple languages and locale conventions. Due to a limitation of Messaging Application Programming Interface (MAPI), however, the rules for encoding the e-mail subject line and the body of the message vary according to the platform. On a UNIX platform, the subject line uses UTF-8 encoding. On a Windows platform, the subject line converts to the code page encoding of the requesting operating system.

If you do not set the content type, the body of the message is plain text. On a Windows platform, the body is inline rich text format (.rtf) text. On a UNIX platform, the body is UTF-8 encoded plain text.

If you specify the content type as plain text, the e-mail message body is plain text. On Windows platforms, characters outside the code page that BIRT iHub uses might not be visible.

Retrieving job properties using GetJobDetails

Use GetJobDetails to retrieve the properties of a job stored in an Encyclopedia volume. An Encyclopedia volume administrator uses GetJobDetails to retrieve the properties of any job in the Encyclopedia volume. A nonadministrative user can use GetJobDetails for a job he submits. If a nonadministrative user sends GetJobDetails for a job he did not submit, BIRT iHub returns a security error.

GetJobDetails retrieves job properties using the elements you specify in the request. The following list describes the available elements:

- JobAttributes returns general job attributes, including JobId, JobName, Priority, Owner, JobType, and InputFileName or ID. JobAttributes always returns in the response to GetJobDetails.
- InputDetail returns details about input parameters.
- Schedules returns schedule information.

- PrinterOptions returns the printer settings, if available.
- NotifyUsers returns the users to notify, if any.
- NotifyGroups returns the groups to notify, if any.
- NotifyChannels returns the channels to notify, if any.
- DefaultOutputFileACL returns the output file ACL templates.
- Status returns the job status.
- Query returns definitions of columns available to this report. It also returns filtering criteria and other information.
- ReportParameters returns the report parameter values. BIRT iHub reads the report parameters from the report object value (.rov) file associated with the job.

The following example shows a response to a GetJobDetails request that includes JobAttributes:

```
<SOAP-ENV:Body>
   <GetJobDetailsResponse>
      <JobAttributes>
         <JobId>58</JobId>
         <JobName>Latest Results</JobName>
         <Priority>1000</Priority>
         <Owner>Administrator</Owner>
         <JobType>RunReport</JobType>
         <State>Succeeded</State>
         <InputFileId>973</InputFileId>
         <InputFileName>/ltor.rptdesign;1</InputFileName>
         <ParameterFileId>1045</ParameterFileId>
         <ParameterFileName>
            /$$$TempROVs/_441c_f0b808e4.ROV;1
         </ParameterFileName>
         <ActualOutputFileId>1046</ActualOutputFileId>
         <ActualOutputFileName>/ltor.rptdocument;26
         </ActualOutputFileName>
         <RequestedOutputFileName>ltor.rptdocument
         </RequestedOutputFileName>
         <SubmissionTime>2008-09-20T21:59:16</SubmissionTime>
         <CompletionTime>2008-09-20T21:59:17</CompletionTime>
         <PageCount>5</PageCount>
         <OutputFileSize>26640</OutputFileSize>
         <RoutedToNode>pinetree</RoutedToNode>
         <DurationSeconds>1</DurationSeconds>
         <StartTime>2008-09-20T21:59:16</StartTime>
         <NotifyCount>1</NotifyCount>
         <RunLatestVersion>false</RunLatestVersion>
      </JobAttributes>
```

```
   </GetJobDetailsResponse>
</SOAP-ENV:Body>
```

Retrieving job properties using GetNoticeJobDetails

When a non-administrative user receives a notification about a job he did not submit, he can use GetNoticeJobDetails to retrieve job details. In addition to the parameters GetJobDetails uses, a GetNoticeJobDetails operation can include the following elements:

- NotifiedChannelId restricts the search to jobs that notify the specified channel ID.
- NotifiedChannelName restricts the search to jobs that notify the specified channel name.

The following GetNoticeJobDetails request uses ResultDef to specify the properties to retrieve and restricts the search to jobs that notify the Managers channel:

```
<SOAP-ENV:Body>
   <GetNoticeJobDetails>
      <JobId>30</JobId>
      <ResultDef>
         <String>InputDetail</String>
         <String>Schedules</String>
         <String>Status</String>
         <String>ReportParameters</String>
      </ResultDef>
      <NotifiedChannelName>Managers</NotifiedChannelName>
   </GetNoticeJobDetails>
</SOAP-ENV:Body>
```

A GetNoticeJobDetails request always returns JobAttributes. The preceding request returns the requested parameters and JobAttributes for jobs that notify the Managers channel.

```
<SOAP-ENV:Body>
   <GetNoticeJobDetailsResponse>
      <JobAttributes>
         <JobId>30</JobId>
         <JobName>Portfolio</JobName>
         <Priority>500</Priority>
         <Owner>Administrator</Owner>
         <JobType>RunReport</JobType>
         …
      </JobAttributes>
```

```
      <InputDetail>
         <OutputMaxVersion>0</OutputMaxVersion>
         <RetryOption>VolumeDefault</RetryOption>
         <MaxRetryCount>10</MaxRetryCount>
         <RetryInterval>6</RetryInterval>
         <MaxVersions>3</MaxVersions>
         <ArchiveOnExpire>false</ArchiveOnExpire>
         …
      </InputDetail>
      <Schedules>
         …
      </Schedules>
      <Status>Starting…</Status>
      <ReportParameters>
         <ParameterValue>
            <Name>RunDate</Name>
            <Value>2008-09-30T00:00:00</Value>
         </ParameterValue>
         …
      </ReportParameters>
   </GetNoticeJobDetailsResponse>
</SOAP-ENV:Body>
```

Cancelling a job

Use CancelJob to stop a print or run request. You can cancel both scheduled and immediate job requests while they are in the Running or Pending state.

Because you identify the job to cancel by its JobId, you must run GetJobDetails if you do not know the JobId.

The following example is a CancelJob request:

```
<SOAP-ENV:Body>
   <CancelJob>
      <JobId>55</JobId>
   </CancelJob>
</SOAP-ENV:Body>
```

CancelJob returns one of the following status messages:

- Succeeded, if the cancellation succeeds
- Failed, if the report completes before or during the cancellation attempt
- InActive, if the job is not in the Running or Pending state

The following example shows a success status message:

```
<SOAP-ENV:Body>
   <CancelJobResponse>
      <Status>Succeeded</Status>
   </CancelJobResponse>
</SOAP-ENV:Body>
```

Working with a resource group

A resource group controls the Factory processes an BIRT iHub uses to run a synchronous or asynchronous job. A resource group specifies a set of Factory processes that execute only jobs assigned to the resource group.

Using the Actuate Information Delivery API, you can perform the following tasks related to resource groups:

- Create, update, and delete a resource group.
- Retrieve a list of resource groups for an BIRT iHub.
- Retrieve details about a specific resource group or all resource groups on an BIRT iHub.
- Reserve a resource group to run specific jobs.
- Enable and disable a resource group.
- Activate a resource group on an BIRT iHub.

The default resource groups are:

- Default Sync runs synchronous jobs.
- Default Async runs asynchronous jobs.

You can create additional resource groups to control processing of specific jobs. When you create a resource group, you define the following properties for it:

- The types of executable files the resource group can run.
- The type of job the resource group can run, either synchronous or asynchronous.
- Whether the resource group is reserved. A reserved resource group runs only the jobs you assign to it using the TargetResourceGroup element in the SOAP header of an ExecuteReport request. You can reserve only a synchronous resource group.
- The priority range of jobs the resource group can run. You can set a priority range only for an asynchronous resource group.
- Whether the resource group is enabled.

- The number of Factory processes assigned to the resource group.
- The BIRT iHub nodes that are members of the resource group.
- The Encyclopedia volumes that can use the resource group's Factory processes. You can assign a resource group to a single Encyclopedia volume in a cluster or to all volumes in the cluster. The Encyclopedia volume to which you assign a resource group does not have to be online, although the resource group will not process jobs until the volume is online.

Creating an asynchronous resource group

To create an asynchronous resource group, you must set the Type element to Async. Use Disabled to enable or disable the resource group.

Use Volume to indicate whether the resource group is assigned to a specific Encyclopedia volume. Valid settings for Volume are an Encyclopedia volume name or an empty string. If you set an Encyclopedia volume name and a request to generate a job comes from a different volume, BIRT iHub rejects the job. If you set an empty string, you assign the resource group to all Encyclopedia volumes on BIRT iHub.

Set a priority range using MinPriority and MaxPriority. The resource group runs only jobs that have priority settings within this range.

If you set a value in Reserved for an asynchronous resource group, BIRT iHub ignores the value. You can reserve only a synchronous resource group.

In ResourceGroupSettings, indicate the name of the BIRT iHub on which the resource group runs. Set Activate to True or False to indicate whether the BIRT iHub is a member of the resource group. Indicate the maximum number of Factory processes to assign to the resource group. List the executable file types the resource group can run.

The following example creates a resource group to run asynchronous jobs from the Corinth Encyclopedia volume, using any one of two executable file types:

```
<SOAP-ENV:Body>
   <CreateResourceGroup>
      <ResourceGroup>
         <Name>End-of-Quarter Sales</Name>
         <Disabled>false</Disabled>
         <Type>async</Type>
         <Volume>Corinth</Volume>
         <MinPriority>500</MinPriority>
         <MaxPriority>1000</MaxPriority>
         <Reserved>false</Reserved>
      </ResourceGroup>
```

```
      <ResourceGroupSettingsList>
         <ResourceGroupSettings>
            <ServerName>Orinda</ServerName>
            <Activate>true</Activate>
            <MaxFactory>2</MaxFactory>
            <FileTypes>
               <String>RPTDESIGN</String>
               <String>RPTDOCUMENT</String>
            </FileTypes>
         </ResourceGroupSettings>
      </ResourceGroupSettingsList>
   </CreateResourceGroup>
</SOAP-ENV:Body>
The response to CreateResourceGroup is an empty operation if the
   job succeeds.
<SOAP-ENV:Body>
   <CreateResourceGroupResponse>
   </CreateResourceGroupResponse>
</SOAP-ENV:Body>
```

If the job fails, an error message appears.

Creating a synchronous resource group

When you create a synchronous resource group, set Type to Sync. If you set priority settings for a synchronous resource group, the Factory ignores them.

Using the Reserved element, you can reserve a synchronous resource group to run only those files assigned to the group in the TargetResourceGroup element of the SOAP header of an ExecuteReport request.

```
<SOAP-ENV:Body>
   <CreateResourceGroup>
      <ResourceGroup>
         <Name>Sales Forecasts</Name>
         <Disabled>false</Disabled>
         <Type>sync</Type>
         <Volume>Fairfield</Volume>
         <Reserved>false</Reserved>
      </ResourceGroup>
   </CreateResourceGroup>
</SOAP-ENV:Body>
```

The response to CreateResourceGroup is an empty operation if the request succeeds. If the request fails, an error message appears.

Updating a resource group's properties

Use UpdateResourceGroup to modify the properties of a single resource group. You can update some, but not all, properties of a resource group using UpdateResourceGroup. For example, you can:

- Enable or disable a resource group.
- Add or modify a description.
- Change the Encyclopedia volume.
- Change the priority range of an asynchronous resource group.
- Change whether the resource group is reserved.
- Update the file types assigned to the resource group.
- Change the setting that indicates whether the BIRT iHub is a member of the resource group.
- Change the maximum number of Factory processes reserved for the resource group.

You cannot update the resource group name or type, or the name of the BIRT iHub on which the resource group runs.

The following example updates a resource group by changing the file types it can run and the maximum number of Factory processes assigned to the resource group:

```
<SOAP-ENV:Body>
   <UpdateResourceGroup>
      <ResourceGroup>
         <Name>End-of-Quarter Sales</Name>
         <Disabled>false</Disabled>
         <Type>async</Type>
         <Volume>Corinth</Volume>
         <MinPriority>500</MinPriority>
         <MaxPriority>1000</MaxPriority>
         <Reserved>false</Reserved>
      </ResourceGroup>
      <ResourceGroupSettingsList>
         <ResourceGroupSettings>
            <ServerName>Orinda</ServerName>
            <Activate>true</Activate>
            <MaxFactory>4</MaxFactory>
            <FileTypes>
               <String>GADGET</String>
            </FileTypes>
```

```
        </ResourceGroupSettings>
      </ResourceGroupSettingsList>
   </UpdateResourceGroup>
</SOAP-ENV:Body>
```

The response to UpdateResourceGroup is an empty operation if the request succeeds. If the request fails, an error message appears.

Getting a list of resource groups

To retrieve a list of resource groups available to BIRT iHub, use GetResourceGroupList, as shown in the following example:

```
<SOAP-ENV:Body>
   <GetResourceGroupList/>
</SOAP-ENV:Body>
```

The preceding request returns two lists, one for asynchronous resource groups and one for synchronous resource groups. Each list contains properties of each resource group, including the name, type, description, whether the resource group is enabled, and whether it is assigned to an Encyclopedia volume.

```
<SOAP-ENV:Body>
   <GetResourceGroupListResponse>
      <AsyncResourceGroupList>
         <ResourceGroup>
            <Name>Default Async</Name>
            <Disabled>false</Disabled>
            <Description>
               Default resource group for asynchronous jobs
            </Description>
            <Type>Async</Type>
            <MinPriority>0</MinPriority>
            <MaxPriority>1000</MaxPriority>
         </ResourceGroup>
      </AsyncResourceGroupList>
      <SyncResourceGroupList>
         <ResourceGroup>
            <Name>Default Sync</Name>
            <Description>Default resource group for synchronous
               jobs</Description>
            <Type>Sync</Type>
            <Reserved>false</Reserved>
         </ResourceGroup>
      </SyncResourceGroupList>
   </GetResourceGroupListResponse>
</SOAP-ENV:Body>
```

Retrieving the properties of a specific resource group

Use GetResourceGroupInfo to retrieve a list of properties for a specific resource group, as shown in the following example:

```
<SOAP-ENV:Body>
   <GetResourceGroupInfo>
      <Name>End-of-Quarter Sales</Name>
   </GetResourceGroupInfo>
</SOAP-ENV:Body>
```

The preceding request returns the properties of the resource group:

```
<SOAP-ENV:Body>
   <GetResourceGroupInfoResponse>
      <ResourceGroup>
         <Name>End-of-Quarter Sales</Name>
         <Disabled>false</Disabled>
         <Type>async</Type>
         <Volume>end00166</Volume>
         <MinPriority>500</MinPriority>
         <MaxPriority>1000</MaxPriority>
      </ResourceGroup>
      <ResourceGroupSettingsList>
         <ResourceGroupSettings>
            <ServerName>end00166</ServerName>
            <Activate>true</Activate>
            <MaxFactory>0</MaxFactory>
            <FileTypes>
               <String>RPTDESIGN</String>
               …
            </FileTypes>
         </ResourceGroupSettings>
      </ResourceGroupSettingsList>
   </GetResourceGroupInfoResponse>
</SOAP-ENV:Body>
```

Retrieving properties for all resource groups on a BIRT iHub

Use GetServerResourceGroupConfiguration to retrieve a list of properties for each resource group on a specific BIRT iHub. The request returns such information as the type of job and the file types each resource group runs, and the maximum Factory processes for each resource group. The Activate element indicates whether BIRT iHub is available to run jobs assigned to the resource group.

```
<SOAP-ENV:Body>
   <GetServerResourceGroupConfiguration>
      <ServerName>end00166</ServerName>
   </GetServerResourceGroupConfiguration>
</SOAP-ENV:Body>
```

The following response shows the settings for the default resource groups:

```
<SOAP-ENV:Body>
   <GetServerResourceGroupConfigurationResponse>
      <ServerResourceGroupSettingList>
         <ServerResourceGroupSetting>
            <ResourceGroupName>Default Async</ResourceGroupName>
            <Type>Async</Type>
            <Activate>true</Activate>
            <MaxFactory>2</MaxFactory>
            <FileTypes>
               <String>RPTDESIGN</String>
               <String>RPTDOCUMENT</String>
            </FileTypes>
         </ServerResourceGroupSetting>
         <ServerResourceGroupSetting>
            <ResourceGroupName>Default Sync</ResourceGroupName>
            <Type>Sync</Type>
            <Activate>true</Activate>
            <MaxFactory>2</MaxFactory>
            <FileTypes>
               <String>RPTDESIGN</String>
               <String>RPTDOCUMENT</String>
            </FileTypes>
         </ServerResourceGroupSetting>
      </ServerResourceGroupSettingList>
   </GetServerResourceGroupConfigurationResponse>
</SOAP-ENV:Body>
```

Setting properties for the resource groups on a BIRT iHub

SetServerResourceGroupConfiguration supports configuring all the resource groups on an BIRT iHub. Use this operation to:

- Change the setting that indicates whether the iHub is a member of the resource group.
- Set or change the maximum number of Factory processes available to the resource group.
- Set or change the file types the resource group can run.

SetServerResourceGroupConfiguration is available only to an Encyclopedia volume administrator or a user in the Administrator role.

```
<SOAP-ENV:Body>
   <SetServerResourceGroupConfiguration>
      <ServerName>end00166</ServerName>
      <ServerResourceGroupSettingList>
         <ServerResourceGroupSetting>
            <ResourceGroupName>Default Sync</ResourceGroupName>
            <Activate>true</Activate>
            <Type>sync</Type>
            <MaxFactory>2</MaxFactory>
            <FileTypes>
               <String>RPX</String>
               …
            </FileTypes>
         </ServerResourceGroupSetting>
         <ServerResourceGroupSetting>
            <ResourceGroupName>Default Async</ResourceGroupName>
            <Activate>true</Activate>
            <Type>async</Type>
            <MaxFactory>2</MaxFactory>
            <FileTypes>
               <String>RPX</String>
            </FileTypes>
         </ServerResourceGroupSetting>
         <ServerResourceGroupSetting>
            <ResourceGroupName>Quarterly Sales
            </ResourceGroupName>
            <Activate>true</Activate>
            <Type>sync</Type>
            <MaxFactory>2</MaxFactory>
            <FileTypes>
               <String>DOX</String>
            </FileTypes>
         </ServerResourceGroupSetting>
         <ServerResourceGroupSetting>
            <ResourceGroupName>Regional Forecasts
            </ResourceGroupName>
            <Activate>true</Activate>
            <Type>sync</Type>
            <MaxFactory>2</MaxFactory>
            <FileTypes>
               <String>RPTDESIGN</String>
               <String>RPTDOCUMENT</String>
            </FileTypes>
         </ServerResourceGroupSetting>
```

```
      </ServerResourceGroupSettingList>
   </SetServerResourceGroupConfiguration>
</SOAP-ENV:Body>
```

The response to SetServerResourceGroupConfiguration is an empty operation if the request succeeds. If the request fails, an error message appears.

Deleting a resource group

To delete a resource group, use DeleteResourceGroup and identify the group to delete, as shown in the following example:

```
<SOAP-ENV:Body>
   <DeleteResourceGroup>
      <Name>End-of-Quarter Sales</Name>
   </DeleteResourceGroup>
</SOAP-ENV:Body>
```

The response to a successful DeleteResourceGroup request is an empty operation if the request succeeds. If the request fails, an error message appears.

Assigning a report to a resource group

You can assign a report to a resource group when you run the report. You must use ExecuteReport and specify the resource group in the optional TargetResourceGroup element of the SOAP header. You can use TargetResourceGroup only to generate a synchronous report.

```
<SOAP-ENV:Header>
   <TargetVolume>end00166</TargetVolume>
   <AuthId>+4yxAKHFJg9FY0JssYijJI5XvkJqDOeA8=</AuthId>
   <TargetResourceGroup>Priority Sync</TargetResourceGroup>
   <Locale>en_US</Locale>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <ExecuteReport>
      <JobName>TRANSIENT_JOB</JobName>
      <InputFileId>90</InputFileId>
      <SaveOutputFile>true</SaveOutputFile>
      <RequestedOutputFile>
         <Name>/mltd.rptdocument</Name>
         <ArchiveRule>
            <FileType>rptdocument</FileType>
         </ArchiveRule>
      </RequestedOutputFile>
      <ProgressiveViewing>true</ProgressiveViewing>
      <WaitTime>20</WaitTime>
```

```
   </ExecuteReport>
</SOAP-ENV:Body>
```

The preceding request returns a ConnectionHandle, the status of the report generation, an object ID, and the output file type.

```
<SOAP-ENV:Header>
   <ConnectionHandle>g7whmBpcJkcHwxpUC6qxppw=
   </ConnectionHandle>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <ExecuteReportResponse>
      <Status>FirstPage</Status>
      <OutputFileType>RPTDOCUMENT</OutputFileType>
      <ObjectId>93</ObjectId>
      <ConnectionHandle>g7whmBpcJcJkcHwxpUC6qxppw=
      </ConnectionHandle>
   </ExecuteReportResponse>
</SOAP-ENV:Body>
```

Assigning a job to a resource group

You can assign a job to a resource group when you submit the job or update a job schedule. Use SubmitJob or UpdateJobSchedule to set properties as you would for any other job. Use the ResourceGroup element to assign the job to a resource group. The following example shows how to set the resource group using SubmitJob:

```
<SOAP-ENV:Body>
   <SubmitJob>
      <JobName>OrderUpdates</JobName>
      <Priority>500</Priority>
      <ResourceGroup>Priority Async</ResourceGroup>
      <InputFileId>90</InputFileId>
      <RunLatestVersion>true</RunLatestVersion>
      <RequestedOutputFile>
         <Name>/OrderUpdates.rptdocument</Name>
      </RequestedOutputFile>
      <Operation>RunReport</Operation>
      <ParameterValues>
         <ParameterValue>
            <Name>NewReportApp::OrderInput::orders_orderID</Name>
            <Value>1645</Value>
         </ParameterValue>
      </ParameterValues>
      <Schedules>
         <TimeZoneName>Pacific Standard Time</TimeZoneName>
```

```
        <ScheduleDetails>
           <JobScheduleDetail>
           …
           </JobScheduleDetail>
        </ScheduleDetails>
     </Schedules>
     <NotifyUsersByName>
        <String>Carl Jacobs</String>
     </NotifyUsersByName>
   </SubmitJob>
</SOAP-ENV:Body>
```

The response to SubmitJob is the JobId.

```
<SOAP-ENV:Body>
   <SubmitJobResponse>
      <JobId>46</JobId>
   </SubmitJobResponse>
</SOAP-ENV:Body>
```

Retrieving the resource group to which a job is assigned

If a user assigns a job to a resource group when submitting the job or updating the schedule, you can determine the resource group using GetJobDetails. Request the resource group in ResultDef, as shown in the following example:

```
<SOAP-ENV:Body>
   <GetJobDetails>
      <JobId>42</JobId>
      <ResultDef>
         <String>Schedules</String>
         <String>ResourceGroup</String>
         <String>InputDetail</String>
      </ResultDef>
   </GetJobDetails>
</SOAP-ENV:Body>
```

If the job is assigned to a resource group, the resource group name appears in JobAttributes in the response.

```
<SOAP-ENV:Body>
   <GetJobDetailsResponse>
      <Schedules>
      …
      </Schedules>
      <InputDetail>
      …
      </InputDetail>
```

```
      <JobAttributes>
         <JobId>42</JobId>
         <JobName>mltd</JobName>
         <Priority>500</Priority>
         <Owner>Administrator</Owner>
         <JobType>RunReport</JobType>
         <State>Scheduled</State>
         …
         <ResourceGroup>Default Async</ResourceGroup>
      </JobAttributes>
   </GetJobDetailsResponse>
</SOAP-ENV:Body>
```

Retrieving and viewing data

The View service supports viewing a document, paging through a document using a web browser, retrieving an entire document or individual components of a document, and viewing the data in a variety of display formats.

The Actuate Information Delivery API supports the View service tasks described in Table 4-1.

Table 4-1 Operations for retrieving and viewing data

Operation	Programming task
GetContent	Retrieving the contents of a report using a component identifier or component name and value. Use the component identifier of 0 to view the entire report.
GetCustomFormat	Retrieving a report in a custom format.
GetEmbeddedComponent	Retrieving an embedded component. The component can be static data, dynamic data, or a style sheet, depending on the suboperation you use.
GetFormats	Retrieving a list of locales or a list of formats that BIRT iHub supports.
GetPageCount	Requesting the number of pages in the report.
GetStyleSheet	Retrieving the style sheet for a report.
GetTOC	Retrieving the table of contents for a report.
SearchReport	Searching a report for specific criteria.
SelectPage	Retrieving a page or range of pages to view.

Typically, a user views a document using Actuate Information Console JavaServer Pages (JSP). Developers can also create custom report viewing pages to integrate with their applications.

Requesting a page or range of pages using SelectPage

Use SelectPage to view a page or range of pages. You must specify an object, a view format, and a page number or page range. Use PageNum to specify a single page or a range of pages. The response returns the requested page or pages as a binary attachment.

The following example shows how to request pages 1 through 6 and page 8. In this request, ViewParameter sets the following conditions:

- The requested display format is DHTML.
- UserAgent is a tool to increase user accessibility to the data.
- AcceptEncoding restricts the content encoding of the response.
- ScalingFactor is set to 100 percent.

```
<SOAP-ENV:Body>
   <SelectPage>
      <Object>
         <Id>1071</Id>
      </Object>
      <ViewParameter>
         <Format>DHTML</Format>
         <UserAgent>Mozilla/4.0 (compatible; MSIE 6.0; Windows
            NT 5.0)
         </UserAgent>
         <AcceptEncoding>gzip, deflate</AcceptEncoding>
         <ScalingFactor>100</ScalingFactor>
      </ViewParameter>
      <Page>
         <PageNum>1-6, 8</PageNum>
      </Page>
   </SelectPage>
</SOAP-ENV:Body>
```

To view the first or the last page of the report, use ViewMode as an element of Page, instead of PageNum. Set ViewMode to 0 to view the first page or 1 to view the last page.

```
<Page>
   <ViewMode>0</ViewMode>
</Page>
```

Retrieving the attachment to a SelectPage response

The response to a SelectPage request includes the SelectPageResponse element, a ConnectionHandle to keep the connection open, a MIME boundary and header, followed by the binary attachment.

In SelectPageResponse, PageRef specifies the following properties of the response:

- ContentId specifies that the pages return as an attachment.
- ContentType is text/html when the requested format is DHTML.
- ContentEncoding is binary, meaning that the pages return as binary data.
- Locale indicates the locale in which the user can view the selected pages.

```
<SOAP-ENV:Envelope>
   <SOAP-ENV:Body>
      <SelectPageResponse>
         <PageRef>
            <ContentId>Attachment</ContentId>
            <ContentType>text/html</ContentType>
            <ContentEncoding>binary</ContentEncoding>
            <Locale>en_US</Locale>
         </PageRef>
         <ConnectionHandle>u7whmBpUho+tg5MUY=</ConnectionHandle>
      </SelectPageResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
67

--MIME_boundary
Content-Type: text/html
Content-Transfer-Encoding:binary
Content-ID:Attachment
806

<HEAD>
   <META NAME="generator" CONTENT="Actuate">
   <META HTTP-EQUIV="Content-Type" CONTENT="text/html;
      charset=UTF-8">
   <LINK REL=STYLESHEET TYPE="text/css"
   …

--MIME_boundary--
0
```

Using SelectPage to print

SelectPage supports printing one or more pages of a report document. In the following example, the ViewOperation element of ViewParameter specifies

printing. SplitOversizePages indicates whether to split a report page across multiple PDF pages. If you set SplitOversizePages to False, each report page prints as a single page. PageWidth and PageHeight indicate the size of the paper.

```
<SOAP-ENV:Body>
   <SelectPage>
      <Object>
         <Id>89</Id>
      </Object>
      <ViewParameter>
         <Format>PDF</Format>
         <UserAgent>
            Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
         </UserAgent>
         <AcceptEncoding>gzip, deflate</AcceptEncoding>
         <ViewOperation>print</ViewOperation>
         <SplitOversizePages>true</SplitOversizePages>
         <PageWidth>6</PageWidth>
         <PageHeight>6</PageHeight>
         <EmbeddedObjPath>ViewEmbeddedObject?
            serverURL=http%3a%2f%2flocalhost%3a4000&amp;
            volume=end00166&amp;
            connectionHandle=g5WwgEamp;operation=</EmbeddedObjPath>
         <RedirectPath>../servlet/GenericRedirector</RedirectPath>
      </ViewParameter>
      <Page>
         <Range>1-5</Range>
      </Page>
   </SelectPage>
</SOAP-ENV:Body>
```

Retrieving report content

Use GetContent to retrieve the contents of a report. You can also retrieve the contents of a component within the report. Using this operation, you specify the report, the format in which to display it, and the component to retrieve from the report. Identify the component by component ID or component name. If you specify a component ID of zero, the response returns the entire report.

You can choose from the following display formats:

- CSS
- DHTML
- DHTMLLong
- DHTMLRaw
- ExcelData

- ExcelDisplay
- ImageMapURL
- PDF
- PPT
- PPTFullyEditable
- Reportlet
- RTF
- RTFFullyEditable
- XMLCompressedDisplay
- XMLCompressedExcel
- XMLCompressedPDF
- XMLCompressedPPT
- XMLCompressedReportlet
- XMLCompressedRTF
- XMLData
- XMLDisplay
- XMLReportlet
- XMLStyle

The PDF format works only for page-based information. A user cannot retrieve component-based data in PDF format.

The Reportlet format works only if the report designer enables ShowInReportlet during report design. If you choose Reportlet, the default value for the maximum height is zero, which means there is no limit to the height of the Reportlet. If you do not specify another value, BIRT iHub converts the whole component into a Reportlet.

The following example requests an entire report in XMLDisplay format. Note that the request includes a MIME boundary.

```
…
--MIME_boundary
Content-Type: text/xml;charset=utf-8
Content-Transfer-Encoding:8bit
Content-ID:<response.xml>
…
<SOAP-ENV:Body>
   <GetContent>
```

```
      <Object>
         <Name>/Temp/forecast.rptdocument</Name>
      </Object>
      <ViewParameter>
         <Format>XMLDisplay</Format>
      </ViewParameter>
      <Component>0</Component>
   </GetContent>
</SOAP-ENV:Body>
```

The preceding request returns the SOAP response and an attachment containing the data. Because the component ID is 0, the attachment contains the entire document.

```
--MIME_boundary
Content-Type: text/xml;charset=utf-8
Content-Transfer-Encoding:8bit
Content-ID:<response.xml>
…
<SOAP-ENV:Body>
   <GetContentResponse>
      <ContentRef>
         <ContentId>Attachment</ContentId>
         <ContentType>text/xml</ContentType>
      </ContentRef>
      <ComponentId>0</ComponentId>
   </GetContentResponse>
</SOAP-ENV:Body>
…
--MIME_boundary
Content-Type: text/xml
Content-Transfer-Encoding:binary
Content-ID:Attachment
800
…

--MIME_boundary
0
```

If the requested display format is Excel, ContentType looks like the following example:

```
<GetContentResponse>
   <ContentRef>
      …
      <ContentType>application/vnd.ms-excel</ContentType>
```

```
      …
   </ContentRef>
   …
</GetContentResponse>
```

Retrieving embedded data and style sheets

Use GetEmbeddedComponent to retrieve:

- Static data, such as an image in a document.
- Dynamic data, such as an embedded URL.
- A style sheet, the template from which Actuate software builds a report. A style sheet returns in cascading style sheets (CSS) format.

Use the Operation element to specify what to retrieve, as shown in the following example:

```
<SOAP-ENV:Body>
   <GetEmbeddedComponent>
      <ObjectId>39</ObjectId>
      <Operation>GetDynamicData</Operation>
      <ComponentId>520</ComponentId>
      <ScalingFactor>57</ScalingFactor>
   </GetEmbeddedComponent>
</SOAP-ENV:Body>
```

This request returns the component as an attachment, along with an EmbeddedRef element showing the type of component. The image follows the response as an attachment.

```
<SOAP-ENV:Body>
   <GetEmbeddedComponentResponse>
      <EmbeddedRef>
         <ContentId>Attachment</ContentId>
         <ContentType>image/jpeg</ContentType>
      </EmbeddedRef>
   </GetEmbeddedComponentResponse>
</SOAP-ENV:Body>
```

Searching within a document

Use SearchReport to request specific criteria within a report. To search a document for a specific component or lists of components, you must specify:

- A target Encyclopedia volume in the SOAP envelope header.
- The name of the document to search.

- The component for which to search. You can search for any report component, including frames, controls, and page lists. Using SearchReportByIdNameList, you can list components by name or ID.
- The value for which to search in each component.
- A view format. Search results return as an attachment. The display format for search results can be in XMLDisplay, tab-separated values, or comma-separated values format.

The following SearchReport request includes all of the required parameters and uses additional parameters in OutputProperties to indicate whether to include column headings in the search result display and whether to enclose each data item in quotation marks:

```
<SOAP-ENV:Body>
   <SearchReport>
      <Object>
         <Name>detail.rptdocument</Name>
      </Object>
      <ViewParameter>
         <Format>XMLDisplay</Format>
         <UserAgent>
            Mozilla/4.0 (compatible; MSIE 5.01; Windows NT)
         </UserAgent>
      </ViewParameter>
      <SearchReportByIdNameList>
         <SearchList>
            <Component>
               <Name>ItemFrame::ItemCategory</Name>
               <Value>Dynamic*</Value>
            </Component>
            <Component>
               <Id>7620</Id>
            <Value>Router</Value>
            <Component>
               <Name>ItemFrame::ItemDescription</Name>
               <Value>16M x 8 Dynamic Ram</Value>
            </Component>
         </SearchList>
         <SelectList>
            <ComponentIdentifier>ItemFrame::ItemCategory
            </ComponentIdentifier>
            <ComponentIdentifier>7620</ComponentIdentifier>
            <ComponentIdentifier>ItemFrame::ItemDescription
            </ComponentIdentifier>
         </SelectList>
```

```
      </SearchReportByIdNameList>
      <OutputProperty>
         <EnableColumnHeaders>true</EnableColumnHeaders>
         <UseQuoteDelimiter>false</UseQuoteDelimiter>
      </OutputProperty>
   </SearchReport>
</SOAP-ENV:Body>
```

The preceding request returns the report or component as an attachment in XMLDisplay. You can also choose Analysis, tab-separated value (TSV), or comma-separated value (CSV) formats.

Searching for a range of pages using SearchReport

The following example uses the Range parameter to indicate a range of search result pages for the response to include. This request searches a range of pages in version 1 of a BIRT document (.rptdocument) file.

```
<SOAP-ENV:Header>
   <TargetVolume>shropshire</TargetVolume>
   <AuthId>G4RhQBq0jidFqdi+o+KJrrSd76mhTYGwG77HJWCj</AuthId>
   <Locale>en_US</Locale>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <SearchReport>
      <Object>
         <Name>/forecast.rptdocument</Name>
         <Version>1</Version>
         <Type>rptdocument</Type>
      </Object>
      <ViewParameter>
         <Format>XMLDisplay</Format>
      </ViewParameter>
      <SelectByIdList>
         <String>674</String>
         <String>660</String>
      </SelectByIdList>
      <SearchByIdList>
         <Component>
            <Id>674</Id>
            <Value>Forecasts</Value>
         </Component>
         <Component>
            <Id>660</Id>
            <Value>Plans</Value>
         </Component>
```

```
      </SearchByIdList>
      <Range>
         <Start>0</Start>
         <End>19</End>
      </Range>
   </SearchReport>
   /SOAP-ENV:Body>
```

In the preceding example:

- TargetVolume is shropshire.
- SearchReport indicates the report to search is version 1 of forecast.rptdocument.
- ViewParameter sets the view format to XMLDisplay.
- Range indicates the range of report pages to search.
- SelectByIDList indicates the component IDs for which to search.
- SearchByIDList defines the values for which to search in each component.

This request returns the component as an attachment, with identifying information about the component:

```
<SOAP-ENV:Header>
   <ConnectionHandle>u5WwgENkpocEYxDO</ConnectionHandle>
</SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <SearchReportResponse>
         <SearchRef>
            <ContentId>Attachment</ContentId>
            <ContentType>text/xml</ContentType>
         </SearchRef>
      </SearchReportResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Header>
```

ConnectionHandle keeps the connection open until the entire response returns.

Getting a table of contents

Use GetTOC to get the table of contents for a report. You must indicate the level from which you want to start viewing and the depth to which you want to view. In Figure 4-1, TocNodeId for Eastern Region Sales Forecast is 0, the top level of the table of contents. Depth indicates how many additional levels you want to retrieve. As shown in Figure 4-1, if you set a depth of 2, the response contains the top level, the first level, showing the offices in the eastern region, and the second level, showing data about sales representatives in each office.

To see the sales representatives' accounts, request a depth of 3.

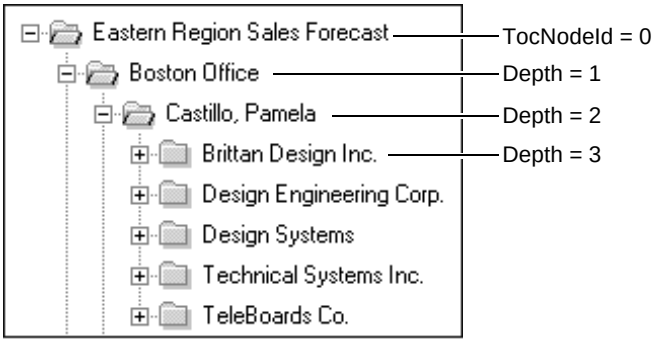


Figure 4-1 Getting a table of contents

The following example requests the table of contents for Forecast.rptdocument:

```
<SOAP-ENV:Body>
   <GetTOC>
      <Object>
         <Name>/Temp/Forecast.rptdocument</Name>
      </Object>
      <ViewParameter>
         <Format>XMLDisplay</Format>
      </ViewParameter>
      <TocNodeId>0</TocNodeId>
      <Depth>2</Depth>
   </GetTOC>
</SOAP-ENV:Body>
```

In this example:

- Object identifies the path to the report.
- ViewParameter indicates the display format, such as XMLDisplay.
- TocNodeId indicates the level from which you want to start viewing.
- Depth indicates the number of levels to retrieve.

This request returns the table of contents as an attachment:

```
<SOAP-ENV:Body>
   <GetTOCResponse>
      <TocRef>
         <ContentId>Attachment</ContentId>
         <ContentType>text/xml</ContentType>
         <ContentEncoding>binary</ContentEncoding>
         <Locale>default</Locale>
      </TocRef>
   </GetTOCResponse>
</SOAP-ENV:Body>
```

Requesting a page count

Use GetPageCount to request a page count for a report. In the request, provide the path to the BIRT document (.rptdocument) file for which you want the count.

```
<SOAP-ENV:Body>
   <GetPageCount>
      <Object>
         <Name>/Temp/Inventory.rptdocument</Name>
      </Object>
   </GetPageCount>
</SOAP-ENV:Body>
```

This request returns the page count, an indicator of whether the report is complete, and a ConnectionHandle in the SOAP header that BIRT iHub uses for subsequent requests for the same report.

```
<SOAP-ENV:Header>
   <AuthId>G4RhQBq0jidFqdi+o+K</AuthId>
   <ConnectionHandle>u7wBuIMEv18lQxjMqWBDBI6/B
   </ConnectionHandle>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <GetPageCountResponse>
      <PageCount>6</PageCount>
      <IsReportCompleted>true</IsReportCompleted>
   </GetPageCountResponse>
</SOAP-ENV:Body>
```

Retrieving display formats

Use GetFormats to retrieve display formats that the View service supports for a report. If you do not specify a format type, GetFormats returns all supported format types for the report. To use GetFormats, specify a path to a BIRT document (.rptdocument) file. You can include other identifying information about the report, such as a version number. The following example shows how to request all formats for a report titled Forecast.rptdocument:

```
<SOAP-ENV:Body>
   <GetFormats>
      <Object>
         <Name>/Temp/Forecast.rptdocument</Name>
         <Version>1</Version>
      </Object>
   </GetFormats>
</SOAP-ENV:Body>
```

The preceding request returns all the formats in which you can view this report, including XMLDisplay, DHTML, Reportlet, cascading style sheets (CSS), ExcelData, and PDF:

```
<SOAP-ENV:Body>
   <GetFormatsResponse>
      <FormatList>
         <Format>XMLDisplay</Format>
         <Format>XMLCompressedDisplay</Format>
         <Format>XMLStyle</Format>
         <Format>XMLData</Format>
         <Format>DHTML</Format>
         <Format>DHTMLLong</Format>
         <Format>DHTMLRaw</Format>
         <Format>Reportlet</Format>
         <Format>CSS</Format>
         <Format>ExcelDisplay</Format>
         <Format>ExcelData</Format>
         <Format>PDF</Format>
      </FormatList>
   </GetFormatsResponse>
</SOAP-ENV:Body>
```

Retrieving a custom format

Use GetCustomFormat to retrieve a report in a custom format from the View service. GetCustomFormat requests that the View service invoke AcReport::GetCustomFormat and then returns the output file as an attachment. The following example uses the ViewParameter element to define the display format:

```
<SOAP-ENV:Body>
   <GetCustomFormat>
      <Object>
         <Name>/CallingBasic.rptdocument</Name>
      </Object>
      <ViewParameter>
         <Format>Excel</Format>
         <UserAgent>
            Mozilla/4.0 (compatible; MSIE 5.01; Windows NT)
         </UserAgent>
         <Locale>default</Locale>
      </ViewParameter>
      <ArgumentList>
         <Argument>
            <Name>case</Name>
```

```
            <Value>Succeeded</Value>
          </Argument>
        </ArgumentList>
    </GetCustomFormat>
</SOAP-ENV:Body>
```

The preceding request returns the document as an attachment in the specified custom format.

Managing a large list

The Actuate Information Delivery API provides the ability to retrieve a large number of objects from a data source. To manage a large list efficiently, the API provides five parameters that provide state information, configure the number of objects to return at one time, and manage the search result in other ways:

- FetchSize indicates the number of records to retrieve and return in the result set at one time. If you do not specify a FetchSize, BIRT iHub uses the default value of 500. If you set FetchSize to 0 or less, no records return.
- FetchHandle returns in a search response when the result set exceeds the FetchSize. FetchHandle returns in the response to a Select or Get request, such as SelectFiles or GetFolderItems. Use FetchHandle to retrieve more results in the set. In the second and subsequent calls for data, you must specify the same search criteria that you used in the original call. All Get and Select requests support FetchHandle except SelectFileType.
- FetchDirection supports navigating through a result set when the results exceed the FetchLimit. FetchDirection supports getting the next or previous set of results in a response when the result set exceeds the FetchSize. To page forward through the result list, set FetchDirection to True. To page in reverse, set FetchDirection to False. The default setting is True. You can set FetchDirection in all Get and Select requests except SelectFileType.

  Differentiate fetch direction from sort order. Fetch direction specifies whether to get the next or previous set of results in a result set. Sort order specifies the order of the data returned by a query. Use the SortOrder attribute in SortColumn and the GroupSortOrder attribute in Grouping to specify the sort order of data returned by a query.
- In Get and Select requests, CountLimit indicates how many objects to count. For example, if the total possible count is 1,000 records, you can limit the count result to the first 100 records. A CountLimit of -1 counts everything in the result set. The default CountLimit is equal to the FetchSize. The count is independent of how many items a response returns.

TotalCount is the response to a CountLimit. In Get and Select responses, TotalCount indicates the size of the counted result set. TotalCount does not return for NameList or IdList requests.

Table 4-2 lists the settings for CountLimit and TotalCount.

Table 4-2 CountLimit and TotalCount settings

Setting	Result
CountLimit = 0	Does not count. Returns a TotalCount of zero.
CountLimit = -1	Counts all objects.
CountLimit > 0	Counts up to the specified limit. In this case, TotalCount returns as the lesser of the CountLimit or the total result set.

The following example shows how these mechanisms appear in the search request. In this example, the result counts up to 600 items, retrieving 100 at a time:

```
<Search>
   <FetchSize>100</FetchSize>
   <FetchDirection>true</FetchDirection>
   <CountLimit>600</CountLimit>
</Search>
```

Working with a large message

The Information Delivery API supports two ways to send and receive a message such as a report response:

- Embed the data in the SOAP body.
- Attach the data to a SOAP request or response.

If you use HTTP 1.0, you typically choose to embed the data in the SOAP message as a single block and send the block in an uninterrupted data stream. If you use HTTP 1.1, you can send the data as an attachment to improve performance on BIRT iHub and the network.

To download or upload a file, you indicate whether to embed the data in the response or send the data as an attachment by setting the DownloadEmbedded

option to True or False. The following example shows a SOAP request to download a file with the content embedded in the body of the response:

```
<SOAP-ENV:Body>
   <DownloadFile xmlns="http://schemas.actuate.com/actuate11">
      <FileName>/report/SampleReport.rptdesign</FileName>
      <DecomposeCompoundDocument>
         false
      </DecomposeCompoundDocument>
      <DownloadEmbedded>true</DownloadEmbedded>
   </DownloadFile>
</SOAP-ENV:Body>
```

The following example shows the SOAP response with the content of the file embedded in the message:

```
<SOAP-ENV:Body
   <ACTU:DownloadFileResponse
      …
      <File>
         <Id>8</Id>
         <Name>/report/SampleReport.rptdesign</Name>
         <FileType>RPTDESIGN</FileType>
         <Version>1</Version>
         <TimeStamp>2008-04-11T19:02:43</TimeStamp>
         <Owner>Administrator</Owner>
         <UserPermissions>VSRWEDG</UserPermissions>
         <PageCount>0</PageCount>
         <Size>47104</Size>
      </File>
         <Content>
            <ContentId>SampleReport.rptdesign</ContentId>
            <ContentType>Application/Octet-Stream</ContentType>
            <ContentData>
               0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAAPg
               …
            </ContentData>
         </Content>
   </ACTU:DownloadFileResponse>
</SOAP-ENV:Body>
```

The following example shows a SOAP request to download a file with the file sent as an attachment:

```
<SOAP-ENV:Body>
   <DownloadFile xmlns="http://schemas.actuate.com/actuate11">
      <FileName>/report/SampleReport.rptdesign</FileName>
      <DecomposeCompoundDocument>
         false
      </DecomposeCompoundDocument>
      <DownloadEmbedded>false</DownloadEmbedded>
   </DownloadFile>
</SOAP-ENV:Body>
```

The following example shows a multi-part response with the MIME boundary and content type defined in the HTTP header and the file placed outside the SOAP envelope as an attachment:

```
HTTP/1.0 200 OK
   Content-Type:
   Multipart/Related;boundary=Mime_boundary;
   type="text/xml";
   start="<response.xml>"HOST:ENL2509Connection:
   closeSOAPAction:
   ""--Mime_boundaryContent-Type: text/xml;
   charset=utf-8Content-Transfer-Encoding:8bitContent-ID:
   <response.xml>
   <?xml version='1.0' ?>
   <SOAP-ENV:Envelope xmlns:SOAP-ENV=
   "http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header
      …
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <ACTU:DownloadFileResponse>
         <File>
            <Id>8</Id>
            <Name>/report/SampleReport.rptdesign</Name>
            <FileType>RPTDESIGN</FileType>
            <Version>1</Version>
            <TimeStamp>2008-04-11T19:02:43</TimeStamp>
            <Owner>Administrator</Owner>
            <UserPermissions>VSRWEDG</UserPermissions>
            <PageCount>0</PageCount>
            <Size>47104</Size>
         </File>
            <Content>
               <ContentId>SampleReport.rptdesign</ContentId>
               <ContentType>
                  Application/Octet-Stream
```

```
            </ContentType>
          </Content>
      </ACTU:DownloadFileResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
   --Mime_boundaryContent-Type:
   Application/Octet-StreamContent-Transfer-Encoding:
   binaryContent-ID:SampleReport.rptdesign
   …
```

Delivering a multilingual document

Actuate software uses the Unicode character set standard to provide multilingual, cross-platform, language-independent reporting. Unicode organizes languages according to locales. A locale is a location plus the language, date and time formats, currency representation, sort order, and other conventions of that location. A locale is not necessarily a country. Using Unicode, Actuate software supports:

- Diacritics such as the tilde (~)
- Composite characters such as ≅
- External font libraries
- Multiple currencies in a single document

Actuate software does not support encoding logos or graphics, font variants, line breaks, or orientation of on-screen characters.

To prompt BIRT iHub to generate a response in the language of that locale, set the Locale parameter in the header of a SOAP envelope. When you do so, the response also uses the date and time formats, currency, and other conventions for that locale. Parameters return in the specified locale. If you specify an invalid locale, the response returns in the default locale, US English. If you do not choose a locale, the response returns in the default locale of the document user's BIRT iHub.

The following SOAP header requests output formatted for the Greek language and locale:

```
<SOAP-ENV:Header>
   <TargetVolume>Grandee</TargetVolume>
   <Locale>el_GR</Locale>
</SOAP-ENV:Header>
```

The response returns document content and parameters formatted for the specified locale, as shown in Figure 4-2.

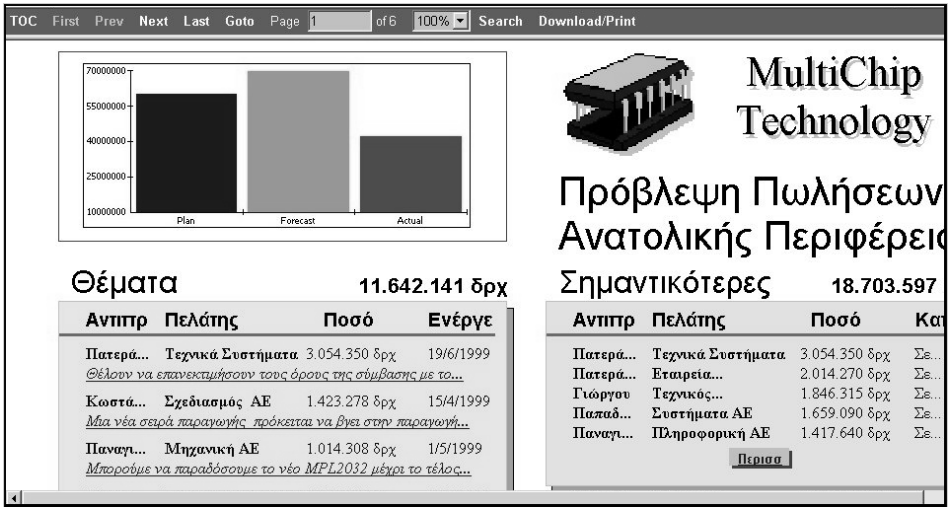


Figure 4-2 Output formatted for the Greek language and locale

Chapter

5

Administering an Encyclopedia volume

This chapter contains the following topics:

- About the Encyclopedia service
- Defining the data on which an operation acts
- Administering security and authentication operations
- About Encyclopedia-level management operations
- Managing Encyclopedia volume items
- About composite operations and transactions
- Searching within an Encyclopedia volume
- Monitoring BIRT iHub information
- Monitoring or cancelling a request for a synchronous report

About the Encyclopedia service

The Encyclopedia service provides the following functional groups of operations:

- Security and authentication operations that manage logging in to a BIRT iHub and authenticating users
- Encyclopedia-level operations such as uploading or downloading files and folders and searching for content across Encyclopedia volumes
- Volume-level operations that manage the items in a particular Encyclopedia volume
- Search operations that retrieve information about items in an Encyclopedia volume

Defining the data on which an operation acts

The Information Delivery API provides three sets of parameters that define the data on which an operation acts. These parameters are:

- Id or IdList
- Name or NameList
- Search

To use Id, IdList, Name, or NameList, define an operation, such as UpdateUser or DeleteGroup, then list each item the operation affects. Identify items by name or BIRT iHub-generated ID number.

Search supports acting on data that meets a specific condition. Typically, these parameters apply to operations that select, update, move, copy, or delete items. The following sections provide examples of using each set of parameters.

Defining data using Id or IdList

To use Id or IdList in an operation, identify the operation. Then, use Id or IdList as a parameter and identify the item on which the operation acts. For example, the following request deletes IdList 4.

IdList 4 is a BIRT iHub-generated identifier for this list.

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
         <DeleteFile>
            <IdList>
               <String>49</String>
```

```
            <String>168</String>
            <String>173</String>
            <String>208</String>
            <String>1067</String>
         </IdList>
      </DeleteFile>
   </AdminOperation>
</Administrate>
</SOAP-ENV:Body>
```

Defining data using Name or NameList

The following code example shows how to define by name the users an UpdateUser operation affects:

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
         <UpdateUser>
            <Name>
               <String>Sam Stein</String>
               <String>Ying Chen</String>
               <String>Helmut Gunther</String>
               <String>Francoise DuBois</String>
            </Name>
         …
         </UpdateUser>
      </AdminOperation>
   </Administrate>
</SOAP-ENV:Body>
```

Defining data using Search

To limit the scope of an operation to specific data, use the Search element to indicate which data the operation affects. The following request deletes all notification groups to which Carl Benning belongs:

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
         <DeleteGroup>
            <Search>
               <WithUserName>Carl Benning</WithUserName>
            </Search>
         </DeleteGroup>
      </AdminOperation>
   </Administrate>
</SOAP-ENV:Body>
```

You cannot delete all items that match a condition by using the wildcard asterisk (*). For example, the following element results in an error message:

```
<WithUserName>*</WithUserName>
```

To delete all items that match a condition, you must list them all.

You can use Search to streamline update and delete requests. For example, to assign UserB the same roles as UserA, use a single UpdateRole operation. First, update the security role by adding UserB, then run a search for UserA in the same request.

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
         <UpdateRole>
            <UpdateRoleOperationGroup>
               <UpdateRoleOperation>
                  <AssignedToUsersByName>
                     <String>UserB</String>
                  </AssignedToUsersByName>
               </UpdateRoleOperation>
               <Search>
                  <UserName>UserA</UserName>
               </Search>
            </UpdateRoleOperationGroup>
         </UpdateRole>
      </AdminOperation>
   </Administrate>
</SOAP-ENV:Body>
```

Administering security and authentication operations

Security and authentication operations govern login and authentication requests. These operations also retrieve access control list (ACL) information for specified files, folders, and channels.

The Actuate Information Delivery API provides two login mechanisms, one for BIRT iHub administrators and one for other users, including volume administrators. The iHub administrator manages iHub and Encyclopedia volumes, performing such tasks as taking an iHub offline and bringing it back online, adding an iHub to a cluster, setting iHub and Encyclopedia volume properties, managing resource groups, and adding printer connections to an iHub.

The Encyclopedia volume administrator controls access to an Encyclopedia volume by creating users and assigning passwords and other credentials. The Encyclopedia volume administrator also uploads and downloads files to and

from the volume and creates, updates, and deletes the items in the Encyclopedia volume.

Other users work with Encyclopedia volume items to the extent that their access privileges permit.

Logging in as a report user

The Login request authenticates a user to BIRT iHub. A request to log in must include the user's login name. It also can include:

- A password or other credentials.
- A domain, the Encyclopedia volume that the user wants to access.
- An indicator of whether to return the user's setting information. The user's settings include such details as the user's name or ID number on BIRT iHub System, default printer, e-mail address, home folder, and viewing preferences. The user's viewing preference can be either DHTML, LRX, or the default setting for the Encyclopedia volume to which the user logs in.
- A list of security roles to validate for the user. You can validate any standard or custom security role by listing the role as a string in ValidateRoles.

The Login response always returns the following elements:

- A required AuthId to authenticate the user to BIRT iHub System. All subsequent requests in the current session must include the AuthId in the SOAP header.
- A list of BIRT iHub options available for the Encyclopedia volume to which the user is logging in.

The Login response also can return:

- The Encyclopedia volume to which the user is logging in. Specify the Encyclopedia volume in the Domain element.
- A list of administrative privileges, if the user is an Encyclopedia volume administrator.
- Details about the user's settings, if you set the optional UserSetting request element to True.
- The user's valid security roles from the list provided in Validate Roles request parameter.

The following example shows a request to log in to the Fairfield Encyclopedia volume using a password. The request asks for user settings and provides a list of security roles to verify.

```
<SOAP-ENV:Body>
   <Login>
      <User>Kevin Belden</User>
      <EncryptedPwd>5hvmuDgEw3E=</EncryptedPwd>
      <Domain>Fairfield</Domain>
      <UserSetting>true</UserSetting>
      <ValidateRoles>
         <String>all</String>
         <String>Active Portal Intermediate</String>
         <String>Regional Managers</String>
      </ValidateRoles>
   </Login>
</SOAP-ENV:Body>
```

The preceding request returns the AuthId, the user's privileges, details about the user, and the BIRT iHub options available. It also returns the security roles that are valid for this user from the list provided in ValidateRoles.

```
<SOAP-ENV:Body>
   <LoginResponse>
      <AuthId>m4yxAKHFdgedY0AlOQBTDAZc==</AuthId>
      <User>
         <Name>Kevin Belden</Name>
         <Id>3022</Id>
         <Description>Southwest Regional Manager</Description>
         <IsLoginDisabled>false</IsLoginDisabled>
         <EmailAddress>kbelden@corpus.com</EmailAddress>
         <HomeFolder>/Belden</HomeFolder>
         <ViewPreference>DHTML</ViewPreference>
         <MaxJobPriority>500</MaxJobPriority>
         <SuccessNoticeExpiration>14400</SuccessNoticeExpiration>
         <FailureNoticeExpiration>14400</FailureNoticeExpiration>
         <SendEmailForSuccess>true</SendEmailForSuccess>
         <AttachReportInEmail>true</AttachReportInEmail>
         <SendNoticeForSuccess>true</SendNoticeForSuccess>
         <SendEmailForFailure>true</SendEmailForFailure>
         <SendNoticeForFailure>true</SendNoticeForFailure>
         <DefaultPrinterName>Sandoval</DefaultPrinterName>
      </User>
      <FeatureOptions>
         <String>ReportGeneration</String>
         <String>PageSecureViewing</String>
      </FeatureOptions>
```

```
      <ValidRoles>
         <String>all</String>
         <String>Regional Managers</String>
      </ValidRoles>
   </LoginResponse>
</SOAP-ENV:Body>
```

Logging in with SystemLogin

SystemLogin authenticates the user as a BIRT iHub system administrator. This login provides access to BIRT iHub system administration functionality, such as managing the properties of a BIRT iHub. To use SystemLogin, you must provide a system password, not your user password.

In Release 10, TargetVolume is an optional element. In Release 11, the Login message must specify the Encyclopedia volume name using TargetVolume in the SOAP header and Domain in the SOAP message body.

```
<SOAP-ENV:Header>
   <TargetVolume>end00166</TargetVolume>
   <Locale>en_US</Locale>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <SystemLogin>
      <SystemPassword>CKJyy769</SystemPassword>
      <SystemPasswordEncryptLevel>1</SystemPasswordEncryptLevel>
   </SystemLogin>
</SOAP-ENV:Body>
```

The response to SystemLogin is an AuthId that remains valid in subsequent requests during the current session.

```
<SOAP-ENV:Body>
   <SystemLoginResponse>
      <AuthId>+zxJgKuMBr4lC1psWCGajW8GXIp7PSC==</AuthId>
   </SystemLoginResponse>
</SOAP-ENV:Body>
```

Getting an access control list

Using the Actuate Information Delivery API, you can retrieve the access control list (ACL) for a file, folder, or channel. You also can retrieve the ACL template that applies to all new files a user creates in an Encyclopedia volume.

Requesting a file or folder's ACL

To retrieve the access rights that apply to a given file or folder, use GetFileACL. Specify the item using its full path.

For a file, you can include a version number, using the format shown in the following example:

```
<SOAP-ENV:Body>
   <GetFileACL>
      <FileName>/Grants and Deeds/Brisbane County Deeds.rptdesign;
         3</FileName>
   </GetFileACL>
</SOAP-ENV:Body>
```

The response returns the privileges for each security role or user that can access the file. The response also includes a TotalCount of users and security roles that can access the file. By default, TotalCount returns 500 records at a time. This limit is configurable. If the list is longer than the limit, the response also includes a FetchHandle mechanism to retrieve the balance of the list.

```
<SOAP-ENV:Body>
   <GetFileACLResponse>
      <ACL>
         <Permission>
            <RoleName>Administrator</RoleName>
            <RoleId>141</RoleId>
            <AccessRight>VSRWEDG</AccessRight>
         </Permission>
         <Permission>
            <RoleName>Visitor</RoleName>
            <RoleId>90</RoleId>
            <AccessRight>V</AccessRight>
         </Permission>
      </ACL>
      <TotalCount>2</TotalCount>
   </GetFileACLResponse>
</SOAP-ENV:Body>
```

Retrieving the ACL for a channel

To retrieve the ACL for a channel, use GetChannelACL. The request must specify the ChannelName or ChannelId, as shown in the following example:

```
<SOAP-ENV:Body>
   <GetChannelACL>
      <ChannelName>BargainBooks</ChannelName>
   </GetChannelACL>
</SOAP-ENV:Body>
```

The preceding request returns the permissions for each security role or user that can access the channel and a TotalCount of the security roles or users that can access it. If the list exceeds the allowable maximum number of items to fetch, the response includes a FetchHandle to support retrieving the remaining items.

```
<SOAP-ENV:Body>
   <GetChannelACLResponse>
      <ACL>
         <Permission>
            <RoleName>Support</RoleName>
            <RoleId>306</RoleId>
            <AccessRight>RW</AccessRight>
         </Permission>
         <Permission>
            <UserName>Raphael Grigory</UserName>
            <UserId>554</UserId>
            <AccessRight>RW</AccessRight>
         </Permission>
      </ACL>
      <TotalCount>2</TotalCount>
   </GetChannelACLResponse>
</SOAP-ENV:Body>
```

Getting a user's ACL template

GetFileCreationACL retrieves a user's ACL template. The ACL template is the access rights that apply to a file the user creates in an Encyclopedia volume. You can specify the user by either ID or name.

```
<SOAP-ENV:Body>
   <GetFileCreationACL>
      <CreatedByUserName>Wenfeng Chan</CreatedByUserName>
   </GetFileCreationACL>
</SOAP-ENV:Body>
```

The response returns the privileges that apply to the user's new files. In the following example, the privileges include visible, read, write, execute, and delete. If the list exceeds the allowable maximum number of items to fetch, the response includes a FetchHandle to support retrieving the remaining items.

```
<SOAP-ENV:Body>
   <GetFileCreationACLResponse>
      <ACL>
         <Permission>
            <AccessRight>VRWED</AccessRight>
         </Permission>
      </ACL>
      <TotalCount>1</TotalCount>
   </GetFileCreationACLResponse>
</SOAP-ENV:Body>
```

Setting an additional condition on an ACL request

Typically, a request for an ACL returns all the permissions that apply to an item. The request returns the permissions for every user and every security role with access to the item. You can, however, restrict a result to the permissions that apply to a specific user or security role.

Using GetFileCreationACL, you can request privileges granted to a user or a security role for files a specific user creates. For example, you can extend the request to ask for the rights granted to the Engineering security role for a file that Wenfeng Chan created. Specify the security role and user by name or ID.

```
<SOAP-ENV:Body>
   <GetFileCreationACL>
      <CreatedByUserName>Wenfeng Chan</CreatedByUserName>
      <GrantedRoleName>Engineering</GrantedRoleName>
   </GetFileCreationACL>
</SOAP-ENV:Body>
```

The preceding request returns the rights granted to Engineering for files Wenfeng Chan creates. In this example, the rights are Visible and Secure Read:

```
<SOAP-ENV:Body>
   <GetFileCreationACLResponse>
      <ACL>
         <Permission>
            <GrantedRoleName>Engineering</GrantedRoleName>
            <AccessRight>VSR</AccessRight>
         </Permission>
      </ACL>
      <TotalCount>1</TotalCount>
   </GetFileCreationACLResponse>
</SOAP-ENV:Body>
```

Using GetChannelACL, you can determine what permissions a user or security role has to a channel by specifying GrantedUserName, GrantedUserId, GrantedRoleName, or GrantedRoleId:

```
<SOAP-ENV:Body>
   <GetChannelACL>
      <ChannelName>BargainBooks</ChannelName>
      <GrantedUserName>Colin Drey</GrantedUserName>
   </GetChannelACL>
</SOAP-ENV:Body>
```

The preceding request returns the privileges for the user or security role and a TotalCount:

```
<SOAP-ENV:Body>
   <GetChannelACLResponse>
      <ACL>
```

```
        <Permission>
          <GrantedUserName>Colin Drey</GrantedUserName>
          <AccessRight>RW</AccessRight>
        </Permission>
      </ACL>
      <TotalCount>1</TotalCount>
    </GetChannelACLResponse>
  </SOAP-ENV:Body>
```

About Encyclopedia-level management operations

Encyclopedia-level management operations support uploading and downloading a file, retrieving a file or folder from an Encyclopedia volume, retrieving an item from a folder, and getting details about a file or folder.

Uploading a file

To upload a file to an Encyclopedia volume, use UploadFile. You must identify the Encyclopedia volume in the SOAP header using TargetVolume. Then, specify the file to upload. You also can set certain properties of the file. For example, you can use the AccessType element to indicate whether the file is private or shared and you can use MaxVersions to set the maximum number of versions of the file to retain in the Encyclopedia volume.

You can upload an Actuate native file type or an external file type. To work with an external file you upload, BIRT iHub must recognize the file type.

Table 5-1 lists the principal elements of an UploadFile request.

Table 5-1 UploadFile request elements

Element	Description
NewFile	Identifies the file to upload. Specify an optional version number by adding it to the file name using a semicolon. In addition to the file name and version, NewFile can contain the following information: ■ AccessType indicates whether the file is shared or private in the Encyclopedia volume. ■ ReplaceExisting indicates whether to replace an existing file that has the same name. If the file you replace has dependencies, BIRT iHub creates a new version and does not overwrite the existing version. ■ MaxVersions indicates the maximum number of versions of the file to retain in the Encyclopedia volume.

(continues)

Table 5-1 UploadFile request elements (continued)

Element	Description
CopyFromLatestVersion	When the Encyclopedia volume contains older versions of the file you are uploading, you can indicate whether to copy certain properties from the latest version of the older file to the newer version. The properties you can copy are: ■ Description, text that describes the file ■ Permissions, the access control list (ACL) specifying the users and roles that can access the file ■ ArchiveRules, rules that control how to age and expire the file If the file to upload already has these properties set, the settings for CopyFromLatestVersion take precedence.
Content	■ ContentType defines the type of file to upload, such as application/octet-stream or binary. ContentType is required. ■ ContentLength specifies the size of the attachment. ContentLength is optional. ■ ContentEncoding indicates the object encoding used, such as binary or application/octet-stream. ContentEncoding is optional. ■ Locale specifies the object locale. Locale is optional. ■ ContentData is the content of the attachment in the format HTTP requires for transmission across the internet.

Uploading a BIRT design

The following request uploads version 2 of a BIRT design file to the target volume specified in the SOAP header. The request keeps a maximum of six versions of the file in the Encyclopedia volume. BIRT iHub does not overwrite existing versions that have the same name as this file. If there are older versions of this file in the Encyclopedia volume, BIRT iHub copies three properties from the latest of those older versions to the new version.

```
<SOAP-ENV:Header>
   <TargetVolume>SeventySix</TargetVolume>
   <AuthId>g4yxAKHFJg9FY0JssYijJI5X=</AuthId>
   <Locale>en_US</Locale>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <UploadFile>
      <NewFile>
         <Name>/Arizona/Phoenix_Q2.rptdesign;2</Name>
         <ReplaceExisting>false</ReplaceExisting>
         <MaxVersions>6</MaxVersions>
      </NewFile>
```

```
      <CopyFromLatestVersion>
         <String>Permissions</String>
         <String>Description</String>
         <String>ArchiveRules</String>
      </CopyFromLatestVersion>
      <Content>
         <ContentId>PQ2.rptdesign</ContentId>
         <ContentType>application/octet-stream</ContentType>
         <ContentLength>4189760</ContentLength>
         <ContentEncoding>utf-8</ContentEncoding>
      </Content>
   </UploadFile>
</SOAP-ENV:Body>
```

The preceding request returns the FileId as a string when the request succeeds, as shown in the following example. If a request fails, an error message appears.

```
<SOAP-ENV:Body>
   <UploadFileResponse>
      <FileId>951</FileId>
   </UploadFileResponse>
</SOAP-ENV:Body>
```

Uploading a third-party report

The following example shows how to upload a Crystal Report (.rpt) file. The request identifies the file to upload using a relative path. It also specifies the content ID and type for the attachment.

```
<SOAP-ENV:Body>
   <UploadFile>
      <NewFile>
         <Name>/report/Phonelist.rpt</Name>
         <ReplaceExisting>true</ReplaceExisting>
      </NewFile>
      <Content>
         <ContentId>Phonelist.rpt</ContentId>
         <ContentType>application/octet-stream</ContentType>
      </Content>
   </UploadFile>
</SOAP-ENV:Body>
```

Copying file properties when uploading a file

Often, when you upload a new version of an executable file to an Encyclopedia volume, the new version must replace the previous version. You must therefore ensure that the permissions and other properties of the previous version apply to the new one. To automate this task, use CopyFromLatestVersion in UploadFile. CopyFromLatestVersion is available whether you replace the existing version or

create a new version during the upload. The following code example shows how to copy two properties, the description and the archive rules, from the previous version of the Sales report:

```
<SOAP-ENV:Body>
   <UploadFile>
      <NewFile>
         <Name>/Reports/Sales.rptdesign</Name>
      </NewFile>
      <CopyFromLatestVersion>
         <String>Description</String>
         <String>ArchiveRules</String>
      </CopyFromLatestVersion>
      <Content>
         <ContentId>Sales.rptdesign</ContentId>
         <ContentType>application/octet-stream</ContentType>
         <ContentEncoding>utf-8</ContentEncoding>
      </Content>
   </UploadFile>
</SOAP-ENV:Body>
```

Attaching or embedding a file in a request

When you upload or download a file, the content streams to or from the Encyclopedia volume using one of the following methods:

- Attach the file in the UploadFile response or DownloadFile request using HTTP chunked transfer encoding. Chunked transfer encoding breaks the data into discrete sections, or chunks, and sends them in a series. This method is useful when the file is long and when BIRT iHub begins sending the response before retrieving all the data. An attachment relies on maintaining a persistent connection, which is the default setting for HTTP 1.1. This method frees BIRT iHub between sections, though the connection remains open. The values used for chunking files must be precise, or the SOAP message may hang.
- Embed the file in the UploadFile response or DownloadFile request and send the message as a single block. Embedding works best with smaller files. To embed a file, specify a ContentLength in the HTTP header instead of chunked transfer encoding. If you use HTTP 1.0, you typically choose to embed the file.

Although BIRT iHub System supports both methods, Actuate operations typically use attachments.

A chunked message consists of three parts:

- An HTTP header
- An Actuate SOAP message such as UploadFile or DownloadFile
- A file to embed or attach

Uploading a file as an attachment

To upload a file as an attachment, the client application must support the following tasks:

- Create the unique, identifying MIME boundary required by the MIME protocol.

  ```
  --MIME_boundary_763bc8438bvc34iyc2mcv
  ```

- Create an HTTP header for the message.
- Create a MIME header for the attachment. A MIME header follows each MIME boundary except the last. The following example shows a typical MIME header:

  ```
  Content-Type: rptdesign
  Content-Transfer-Encoding: binary
  Content-ID: Forecast.rptdesign
  ```

  where

  - Content-Type is the type of file to upload. Content-Type is an optional element.
  - Content-ID and Content-Transfer-Encoding are required elements. The Content-ID in the MIME header maps to the ContentId element of the UploadFile operation.

- Prepare and send the SOAP request. An UploadFile request must include the Content element.
- Send the attachment.
- End the message with a final MIME boundary followed by a zero (0).

  ```
  --MIME_boundary_gwt[heqhypodjh;
  0
  ```

About the HTTP header

A typical HTTP header that contains a MIME boundary looks like the following example:

```
POST / HTTP/1.1
Host: akiko:9000
Content-Type: multipart/related;
   boundary=MIME_boundary_763bc8438bvc34iyc2mcv;
type=text/xml; start=<request.xml>
Transfer-Encoding:chunked
MIME-Version:1.0
SOAPAction: ""
EA
```

where

- POST/HTTP/1.1 is a required element that indicates the version of HTTP the message uses.
- Host: akiko:9000 is the name and port number of the host machine.
- Content-Type: multipart/related is a required element.
- boundary=MIME_boundary_763bc8438bvc34iyc2mcv is the MIME boundary.
- start=<request.xml> is a required element that refers to the Content-ID in the first MIME heading.
- Transfer-Encoding is a required element to inform BIRT iHub that this is a chunked message.
- MIME-Version is the version of MIME the message uses.
- SOAPAction is a required element that the Actuate Information Delivery API does not use. Represent SOAPAction by a set of quotation marks.
- EA is a hexadecimal value that represents the size of the chunk.

The client application creates MIME boundaries randomly to ensure the message's uniqueness.

Writing the UploadFile request

The UploadFile request follows the first MIME boundary. The following example requests that the Encyclopedia volume upload one version of a file titled Forecast.rptdesign:

```
<SOAP-ENV:Envelope>
   <SOAP-ENV:Header>
      <AuthId>8ywJQHtcZreNuzqEUboyrIU=</AuthId>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <UploadFile>
         <NewFile>
            <Name>Forecast.rptdesign</Name>
            <MaxVersions>1</MaxVersions>
         </NewFile>
         <Content>
            <ContentId>Forecast.rptdesign</ContentId>
            <ContentType>application/octet-stream</ContentType>
            <ContentLength>true</ContentLength>
            <ContentEncoding>utf-8</ContentEncoding>
         </Content>
      </UploadFile>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
5a

--MIME_boundary
Content-Transfer-Encoding:binary
Content-ID:MultiSetTimeSeries.rptdesign
…

--MIME_boundary--
0
```

The response to an UploadFile request is an identifier for the file or folder.

```
<SOAP-ENV:Body>
   <UploadFileResponse>
      <FileId>25</FileId>
   </UploadFileResponse>
</SOAP-ENV:Body>.
```

Downloading a file

To download a file from an Encyclopedia volume, use DownloadFile and specify the path and either a FileName or FileId, as shown in the following example:

```
<SOAP-ENV:Body>
   <DownloadFile>
      <FileName>/Inventory/FallPromo.rptdesign;2</FileName>
   </DownloadFile>
</SOAP-ENV:Body>
```

The file streams to the client as an attachment to the response, which includes identifying information about the file, such as the file type, owner, length, version, page count, and content ID and type. The ContentId element maps to the Content_ID in the MIME header of the attachment.

```
<SOAP-ENV:Body>
   <DownloadFileResponse>
      <File>
         <Id>949</Id>
         <Name>/Marketing/FallPromo.rptdesign</Name>
         <FileType>RPTDESIGN</FileType>
         <TimeStamp>2008-03-06T21:58:05</TimeStamp>
         <Owner>Craig Lew</Owner>
         <UserPermissions>VSRWEDG</UserPermissions>
         <Version>2</Version>
         <PageCount>6</PageCount>
         <Size>38912</Size>
      </File>
      <Content>
         <ContentId>598</ContentId>
```

```
        <ContentType>application/octet-stream</ContentType>
      </Content>
   </DownloadFileResponse>
</SOAP-ENV:Body>
```

Downloading a file as an attachment

When BIRT iHub receives a DownloadFile request, it creates an HTTP header for the response. This header specifies the multipart-related content type and the chunked transfer-encoding method. The client application must be able to process the information in the header.

Following the HTTP header, BIRT iHub sends a DownloadFile response that includes a Content element. ContentId and ContentType in this element map to Content-ID and Content-Type in the MIME header of the attachment that follows the response.

```
<SOAP-ENV:Body>
   <DownloadFileResponse>
      <File>
         …
      </File>
      <Content>
         <ContentId>598</ContentId>
         <ContentType>application/octet-stream</ContentType>
      </Content>
   </DownloadFileResponse>
</SOAP-ENV:Body>
```

The requested file content streams to the client as an attachment. BIRT iHub creates MIME boundaries randomly to ensure that the message is unique.

Updating a file or folder

You can update a file or folder to create or remove a dependency, set or remove an archive rule, define a parameter, or change other properties. You can also grant or revoke a permission and copy the access control list (ACL) of a folder to its subfolders. To use UpdateFile to modify the properties of a file or folder, you must have the write privilege on the file or folder. To update privileges to the file or folder, you must have the grant privilege on the file or folder.

The following request modifies the autoarchive rules of three files:

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
         <UpdateFile>
            <WorkingFolderId>170</WorkingFolderId>
            <UpdateFileOperationGroup>
```

```
            <UpdateFileOperation>
               <SetArchiveRules>
                  <ArchiveRule>
                     <FileType>RPTDESIGN</FileType>
                     <NeverExpire>false</NeverExpire>
                     <ArchiveOnExpiration>true
                     </ArchiveOnExpiration>
                     <ExpirationAge>86400</ExpirationAge>
                  </ArchiveRule>
               </SetArchiveRules>
            </UpdateFileOperation>
         </UpdateFileOperationGroup>
         <IdList>
            <String>1570</String>
            <String>7840</String>
            <String>8621</String>
         </IdList>
      </UpdateFile>
   </AdminOperation>
</Administrate>
</SOAP-ENV:Body>
```

The response to an UpdateFile request is an empty response if the request succeeds. If the request fails, an error message appears.

Updating a file's parameters

The following example updates a document by setting the ad hoc parameter AC_KEEP_WORKSPACE_DIRECTORY to Yes. This setting preserves the workspace directory after the document runs. Because IsAdHoc is True, ColumnName and ColumnType are required parameters. They apply only to ad hoc parameter definitions. ColumnName is the database column to which the parameter applies. ColumnType is the data type of the column. ColumnType can be any Actuate data type, such as Boolean, Double, and Integer. ColumnName and ColumnType are parameters of ParameterDefinition.

```
<Administrate>
   <AdminOperation>
      <UpdateFile>
         …
            <SetParameterDefinitions>
               <ParameterDefinition>
                  <Name>AC_KEEP_WORKSPACE_DIRECTORY</Name>
                  <DataType>String</DataType>
                  <DefaultValue>Yes</DefaultValue>
                  <IsRequired>false</IsRequired>
                  <IsPassword>false</IsPassword>
                  <IsHidden>false</IsHidden>
```

```
                <IsAdHoc>true</IsAdHoc>
                <ColumnName>KeepW</ColumnName>
                <ColumnType>String</ColumnType>
             </ParameterDefinition>
          </SetParameterDefinitions>
       …
       <Id>467</Id>
    </UpdateFile
  </AdminOperation>
</Administrate>
```

Updating the privilege settings of a file or folder

Using UpdateFile, you can apply the ACL of a folder to files and folders within it. Depending on the suboperation you use, you can either replace an object's ACL with those of the folder that contains it or add a folder's privileges to those of an object in the folder.

For example, the ACL of the Marketing folder grants the Buyer security role visible privileges to Marketing. The ACL of a subdirectory within the Marketing folder grants the Buyer security role read and write privileges. Adding the ACLs of Marketing and the subdirectory within Marketing gives the Buyer security role visible, read, and write privileges to the subdirectory. On the other hand, if you replace the ACL of the subdirectory, the Buyer security role has only visible privileges to the subdirectory.

Whether you add or replace an ACL, you can choose to make the results of the operation recursive. Using the recursive option, you can apply the results to all files and folders in the working folder, including all files and folders in a subdirectory.

Replacing an object's ACL with that of the folder that contains it

To replace the privileges of a file or folder with those of the folder that contains it, use the SetPermissions suboperation of UpdateFile, as shown in the following example. In this example, the AccessRight element for each security role or user defines the security role or user's privileges to the working folder. Use an empty Search element to replace the ACL of all files and folders in the working folder. The omission of the Recursive element indicates that the operation affects only the immediate descendants of the working folder.

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
         <UpdateFile>
            <WorkingFolderId>170</WorkingFolderId>
            <UpdateFileOperationGroup>
               <UpdateFileOperation>
                  <SetPermissions>
```

```
                  <Permission>
                     <RoleName>Regional Managers</RoleName>
                     <AccessRight>S</AccessRight>
                  </Permission>
                  <Permission>
                     <RoleName>Operator</RoleName>
                     <AccessRight>VSREWDG</AccessRight>
                  </Permission>
                  <Permission>
                     <RoleName>Active Portal Administrator
                     </RoleName>
                     <AccessRight>VSREWDG</AccessRight>
                  </Permission>
               </SetPermissions>
            </UpdateFileOperation>
         </UpdateFileOperationGroup>
         <Search/>
      </UpdateFile>
   </AdminOperation>
</Administrate>
</SOAP-ENV:Body>
```

Adding the ACL of a folder to an object in the folder

To add the ACL of a folder to that of an object in the folder, use the GrantPermissions suboperation of UpdateFile. In the following example, access rights to the working folder for two security roles and a user appear in the AccessRight element. The result of this operation is that these security roles and this user receive the same rights to files in the working folder.

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
         <UpdateFile>
            <WorkingFolderName>/Sales/Eastern Region
            </WorkingFolderName>
            <UpdateFileOperationGroup>
               <UpdateFileOperation>
                  <GrantPermissions>
                     <Permission>
                        <RoleName>Sales</RoleName>
                        <AccessRight>SVE</AccessRight>
                     </Permission>
                     <Permission>
                        <RoleName>Eastern Sales Managers
                        </RoleName>
                        <AccessRight>VSREWDG</AccessRight>
                     </Permission>
```

```
                    <Permission>
                       <UserName>Kevan Blaine</UserName>
                       <AccessRight>VSREWDG</AccessRight>
                    </Permission>
                 </GrantPermissions>
              </UpdateFileOperation>
           </UpdateFileOperationGroup>
           <Search/>
        </UpdateFile>
     </AdminOperation>
   </Administrate>
</SOAP-ENV:Body>
```

Use an empty Search element to update the ACL of all files and folders in the working folder. The omission of the Recursive element indicates that this operation affects only the immediate descendants of the working folder.

Adding privileges recursively

To add privileges from a directory to all files and folders within it, including subfolders and their contents, set the Recursive element to True.

```
<UpdateFile>
   <WorkingFolderName>/Sales/Eastern Region</WorkingFolderName>
   <Recursive>True</Recursive>
   …
</UpdateFile>
```

Selecting properties of a file or folder in an Encyclopedia volume

Use SelectFiles to retrieve the name or ID of a single file or folder or a list of files or folders in an Encyclopedia volume. Using ResultDef, you can also retrieve specific properties of each file or folder or list. SelectFiles does not retrieve file or folder content.

To select a file or folder and view its properties, use TargetVolume in the SOAP header to indicate which Encyclopedia volume contains the item. SelectFiles can search all subfolders in a directory. To restrict a search to the items in a single directory, not including subfolders, use GetFolderItems.

SelectFiles supports searches that use the following criteria:

- Use NameList or IdList to retrieve a list of files or folders in a directory.
- Use Name or Id to retrieve a single file or folder.
- Use Search to retrieve all files or folders that match a given condition.

In a SelectFiles request, you can identify by name or ID the working folder from which to select files. You can indicate whether the search is recursive, meaning

that it includes subdirectories of the working folder. You can also specify whether to retrieve only the latest version of the file.

Use ResultDef to specify the file or folder properties to retrieve. The properties you set in ResultDef depend on the type of item. For example, if the item is a file, you can retrieve the file name, ID, description, file type, page count, date of creation or last update, owner, and version name and number. If the item is a folder, you cannot retrieve a file type, version name and number, file size, or page count.

Using SelectFiles, you can set a privilege filter that restricts the result to users in a given security role. When the results of a SelectFiles request exceed the FetchSize, the response returns a FetchHandle to support retrieving the total result.

Requesting a list of files or folders in a working directory using SelectFiles

The following request asks for a list of files and folders in the working directory that match certain criteria. ResultDef specifies the information to return about each file. The request in this example asks for items such as the file ID, name, file type, and version number.

The Recursive element directs BIRT iHub to search subdirectories of the working folder. Search sets the FetchSize and CountLimit.

```
<SOAP-ENV:Body>
   <SelectFiles>
      <WorkingFolderName>/</WorkingFolderName>
      <Recursive>true</Recursive>
      <LatestVersionOnly>true</LatestVersionOnly>
      <ResultDef>
         <String>Id</String>
         <String>FileType</String>
         <String>Version</String>
         <String>Name</String>
         <String>Size</String>
         <String>PageCount</String>
      </ResultDef>
      <Search>
         <FetchSize>500</FetchSize>
         <CountLimit>530</CountLimit>
      </Search>
   </SelectFiles>
</SOAP-ENV:Body>
```

The response to the preceding request is a list of files that match the search conditions. For each file, the response displays the information requested in ResultDef. If the item is a folder, Version, Size, and PageCount return zero (0).

```
<SOAP-ENV:Body>
   <SelectFilesResponse>
      <ItemList>
         <File>
            <Id>1</Id>
            <Name>/</Name>
            <FileType>Directory</FileType>
            <Version>0</Version>
            <Size>0</Size>
            <PageCount>0</PageCount>
         </File>
         <File>
            <Id>11</Id>
            <Name>/Regional Forecasts/mltd.rptdocument</Name>
            <FileType>RPTDOCUMENT</FileType>
            <Version>1</Version>
            <Size>19594</Size>
            <PageCount>6</PageCount>
         </File>
         …
      </ItemList>
      <TotalCount>38</TotalCount>
   </SelectFilesResponse>
</SOAP-ENV:Body
```

You can use file IDs in subsequent requests to move, copy, update, delete, or work in another way with one or more of these files.

About the Search element in SelectFiles

You can refine a SelectFiles search by adding criteria in the Search element. Use the Condition element to define the type of files to select. Add AccessType to the Search element to select private or shared files.

```
<SOAP-ENV:Body>
   <SelectFiles>
      <WorkingFolderName>/</WorkingFolderName>
      <Recursive>false</Recursive>
      <LatestVersionOnly>true</LatestVersionOnly>
      <ResultDef>
         <String>Id</String>
         <String>FileType</String>
         <String>Version</String>
         <String>Name</String>
         <String>VersionName</String>
         <String>Size</String>
         <String>PageCount</String>
      </ResultDef>
```

```
      <Search>
         <Condition>
            <Field>FileType</Field>
            <Match>Directory</Match>
         </Condition>
         <FetchSize>500</FetchSize>
         <CountLimit>1500</CountLimit>
         <AccessType>Private</AccessType>
      </Search>
   </SelectFiles>
</SOAP-ENV:Body>
```

For a folder, Version, Size, and PageCount always return zero (0) in SelectFiles:

```
<SOAP-ENV:Body>
   <SelectFilesResponse>
      <ItemList>
         <File>
            <Id>7</Id>
            <Name>/Queries</Name>
            <FileType>Directory</FileType>
            <Version>0</Version>
            <Size>0</Size>
            <PageCount>0</PageCount>
         </File>
         <File>
            <Id>3</Id>
            <Name>/Requirements</Name>
            <FileType>Directory</FileType>
            <Version>0</Version>
            <Size>0</Size>
            <PageCount>0</PageCount>
         </File>
      </ItemList>
      <TotalCount>2</TotalCount>
   </SelectFilesResponse>
</SOAP-ENV:Body>
```

Using a privilege filter with SelectFiles

A privilege filter ensures that the response displays only the data accessible to users in a given security role. You can use a privilege filter with SelectFiles. The following example requests every file in the Sales directory that is accessible to users in the Manager role who have read and write privileges:

```
<SOAP-ENV:Body>
   <SelectFiles>
      <WorkingFolderName>/Sales</WorkingFoderName>
      <ResultDef>
         <String>Id</String>
         <String>Name</String>
         <String>PageCount</String>
         <String>Size</String>
         <String>TimeStamp</String>
         <String>Owner</String>
      </ResultDef>
      <Search>
         <PrivilegeFilter>
            <GrantedRoleName>Manager</GrantedRoleName>
            <AccessRights>RW</AccessRights>
         </PrivilegeFilter>
      </Search>
   </SelectFiles>
</SOAP-ENV:Body>
```

Retrieving a property list for an item in a folder

GetFolderItems retrieves a list of files or folders in an Encyclopedia volume folder. It also retrieves the properties of those files or folders. To use GetFolderItems, specify the name of the folder to search and use ResultDef to indicate which properties to return for each item.

The properties you can specify include:

- Name
- ID
- FileType
- Description
- PageCount
- Size
- TimeStamp of the most recent update
- Version
- VersionName
- Owner
- User privileges associated with the item

The response always includes the item name and ID.

If a request includes TimeStamp, the application accessing the time stamp must correct for the local time zone. GetFolderItems does not return item content.

Retrieving properties of an item in a folder

The following request looks for every item in the TimeStudy folder and asks for the name, ID, size, and version of each. To identify the folder, you must use a path.

```
<SOAP-ENV:Body>
   <GetFolderItems>
      <FolderName>/TimeStudy</FolderName>
      <ResultDef>
         <String>Name</String>
         <String>Id</String>
         <String>Size</String>
         <String>Version</String>
      </ResultDef>
   </GetFolderItems>
</SOAP-ENV:Body>
```

The preceding request returns the requested properties for each item in the folder and a total count of items returned.

```
<SOAP-ENV:Body>
   <GetFolderItemsResponse>
      <Files>
         <File>
            <Name>Engineering.rptdesign</Name>
            <Id>642</Id>
            <Version>1</Version>
            <Size>38912</Size>
         </File>
         <File>
            <Name>chart.gif</Name>
            <Id>879</Id>
            <Version>6</Version>
            <Size>38912</Size>
         </File>
         <File>
            <Id>1380</Id>
            <Name>Research.rptdesign</Name>
            <Version>3</Version>
            <Size>38912</Size>
         </File>
      </Files>
      <TotalCount>3</TotalCount>
   </GetFolderItemsResponse>
</SOAP-ENV:Body>
```

Setting a condition using GetFolderItems

To limit the search of a folder to those items that match a specific condition, use GetFolderItems and identify the condition in the Search element. Use ResultDef to specify the information to return about each item. For example, to search only for folders in the working directory, send the following request:

```
<SOAP-ENV:Body>
   <GetFolderItems>
      <FolderName>/</FolderName>
      <ResultDef>
         <String>Name</String>
         <String>FileType</String>
      </ResultDef>
      <Search>
         <Condition>
            <Field>FileType</Field>
            <Match>Directory</Match>
         </Condition>
         <FetchSize>100</FetchSize>
         <CountLimit>150</CountLimit>
      </Search>
   </GetFolderItems>
</SOAP-ENV:Body>
```

The preceding request returns the file name and file type of each directory in the working folder. It also returns an identifier for each file and a count of the items.

```
<SOAP-ENV:Body>
   <GetFolderItemsResponse>
      <ItemList>
         <File>
            <Id>14</Id>
            <Name>Belden</Name>
            <FileType>Directory</FileType>
         </File>
         <File>
            <Id>31</Id>
            <Name>Brothers</Name>
            <FileType>Directory</FileType>
         </File>
         …
      </ItemList>
      <TotalCount>10</TotalCount>
   </GetFolderItemsResponse>
</SOAP-ENV:Body>
```

Using GetFileDetails

GetFileDetails retrieves properties of a file or folder. For example, you can retrieve the access control list (ACL) or autoarchive rules of a BIRT or third-party report. Using the AccessType element in ResultDef, you can determine whether the file is private or shared. You also can retrieve information about file dependencies, file ownership, and the size of the file.

Getting the details of an Actuate report

The following request asks for the ACL, autoarchive rules, dependent files, and required files of a file identified by its ID number:

```
<SOAP-ENV:Body>
   <GetFileDetails>
      <FileId>146</FileId>
      <ResultDef>
         <String>ACL</String>
         <String>ArchiveRules</String>
      </ResultDef>
   </GetFileDetails>
</SOAP-ENV:Body>
```

Unlike other Get and Select operations, GetFileDetails returns all the properties for a file, even if you request specific properties.

```
<SOAP-ENV:Body>
   <GetFileDetailsResponse>
      <ArchiveRules>
         <File>
            <Id>146</Id>
            <Name>/CustomerOrders.rptdesign</Name>
            <FileType>RPTDESIGN</FileType>
            <Version>1</Version>
            <TimeStamp>2008-11-10T19:24:08</TimeStamp>
            <Owner>Administrator</Owner>
            <UserPermissions>VSRWEDG</UserPermissions>
            <PageCount>300</PageCount>
            <Size>5427267</Size>
         </File>
         <ACL>
            <Permission>
               <RoleName>Regional Managers</RoleName>
               <AccessRight>VSRWEDG</AccessRight>
            </Permission>
            <Permission>
               <UserName>Kevin Belden</UserName>
               <AccessRight>S</AccessRight>
            </Permission>
```

```
        </ACL>
      </ArchiveRules>
   </GetFileDetailsResponse>
</SOAP-ENV:Body>
```

Managing Encyclopedia volume items

The Actuate Information Delivery API supports managing the items in an Encyclopedia volume and updating volume properties using the Administrate element. Administrate is not an operation on its own. It is a grouping mechanism for the types of operations that an administrator performs. With these operations, an Encyclopedia volume administrator manages the users, groups, roles, channels, file types, files, folders, jobs, and job schedules in an Encyclopedia volume. The administrator also can set or update the properties of the Encyclopedia volume.

You can create a composite operation using the Administrate element. A composite operation accomplishes several tasks in a single operation. In general, the operations you can group into a composite operation are the Create, Update, and Delete operations. Many Administrate operations do not require SOAP responses unless all or part of the operation fails.

Ignoring error conditions in an Administrate operation

An Administrate operation provides two optional request elements that tell BIRT iHub to ignore any error condition it finds and continue the operation. These elements are IgnoreMissing and IgnoreDup. An Ignore element typically applies to the Name or NameList element of the primary object.

To ignore the error condition that BIRT iHub cannot find a specified object, set IgnoreMissing to True. For example, in an UpdateUser operation, if you specify an invalid user name, BIRT iHub ignores the missing name and the operation continues when IgnoreMissing is True. If you set IgnoreMissing to False, the operation stops and returns an error at the missing name.

To ignore the error condition that a specified object already exists, set IgnoreDup to True. BIRT iHub always rejects a duplicate request, regardless of the IgnoreDup setting. When you set IgnoreDup, you indicate whether the operation stops and BIRT iHub returns an error. BIRT iHub does not honor IgnoreDup if you attempt to update multiple objects that have the same name.

Creating an item in an Encyclopedia volume

To create an item such as a user, a folder, a security role, or a notification group in an Encyclopedia volume, you typically indicate the type of item to create, then

create properties for it. The type of item you are creating determines the properties you create.

Creating a user

The following request creates three new users in the Encyclopedia volume and identifies them by various properties. To add more than one e-mail address for a user, separate the addresses with a comma. To refer to the user's home folder, include a complete path, as shown in the following request:

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
         <CreateUser>
            <User>
               <Name>Akiko Takagishi</Name>
               <Password>movies</Password>
               <EmailAddress>ATaka@grisworld.com</EmailAddress>
            </User>
            <User>
               <Name>Grandford Lynne</Name>
               <Password>hedge</Password>
            </User>
            <User>
               <Name>Sandy Browne</Name>
               <Password>nobhill</Password>
               <HomeFolder>/Financials/Sandy</HomeFolder>
            </User>
         </CreateUser>
      </AdminOperation>
   </Administrate>
</SOAP-ENV:Body>
```

Creating a folder

Many operations that create an item must be accomplished using multiple operations. The following example shows how to use CreateFolder to create a folder in the working directory. Then, it uses UpdateFile to set the access type and autoarchive rules. AccessType defines whether the folder is shared or private. ArchiveRule defines when and how to archive the folder. In ArchiveRule, the file type $$$ indicates a folder.

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
         <Transaction>
            <TransactionOperation>
```

```
            <CreateFolder>
               <FolderName>/Requirements</FolderName>
               <IgnoreDup>false</IgnoreDup>
            </CreateFolder>
            <UpdateFile>
               <SetAttributes>
                  <AccessType>Private</AccessType>
               </SetAttributes>
               <SetArchiveRules>
                  <ArchiveRule>
                     <FileType>$$$</FileType>
                     <NeverExpire>false</NeverExpire>
                     <ArchiveOnExpiration>true
                     </ArchivnExpiration>
                     <ExpirationAge>86400</ExpirationAge>
                     <IsInherited>false</IsInherited>
                  </ArchiveRule>
               </SetArchiveRules>
               <Name>/Requirements</Name>
            </UpdateFile>
         </TransactionOperation>
      </Transaction>
   </AdminOperation>
</Administrate>
</SOAP-ENV:Body>
```

The response to a successful CreateFolder request is an empty Administrate response if the request succeeds. If the request fails, an error message appears.

Creating a security role

To create a security role, you assign the role a name, then update it by choosing a parent or child role, and setting privileges.

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
         <Transaction>
            <TransactionOperation>
               <CreateRole>
                  <Role>
                     <Name>Field Sales</Name>
                  </Role>
                  <IgnoreDup>false</IgnoreDup>
               </CreateRole>
               <UpdateRole>
                  <SetParentRolesByName>
                     <String>Sales Representatives</String>
```

```
                </SetParentRolesByName>
                <NameList>
                   <Name>Field Sales</Name>
                </NameList>
             </UpdateRole>
          </TransactionOperation>
       </Transaction>
    </AdminOperation>
  </Administrate>
</SOAP-ENV:Body>
```

The response to a successful CreateRole request is an empty Administrate response. If the request fails, an error message appears.

Deleting an item

The Actuate Information Delivery API supports deleting items from an Encyclopedia volume, such as a file, a user, a folder, a group, a job schedule, or a job notice.

There are three types of Delete operations:

- Delete a single item identified by Name or Id.
- Delete a list of items identified by NameList or IdList.
- Delete all items that match a given condition using Search.

For some Delete operations, you can indicate whether a deletion is recursive, meaning it applies to all items within the item to delete. For example, if DeleteFile is recursive and the file is a folder, the operation deletes the folder and any files and subfolders within it.

A Delete operation returns a status message when it completes successfully or stops at the first failed operation.

Deleting a file or folder

Use DeleteFile to delete a file or folder in an Encyclopedia volume. To delete lists of items, use IdList and specify the lists to delete, as shown in the following example:

```
<SOAP-ENV:Body>
  <Administrate>
    <AdminOperation>
      <DeleteFile>
        <IdList>
          <String>3</String>
          <String>760>/String>
          <String>814</String>
        </IdList>
```

```
        </DeleteFile>
      </AdminOperation>
   </Administrate>
</SOAP-ENV:Body>
```

Deleting a user

The following DeleteUser request is a transaction, which means that all deletions in the request must succeed for any deletions to occur.

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
         <Transaction>
            <TransactionOperation>
               <DeleteUser>
                  <IdList>
                     <String>5</String>
                     <String>461</String>
                     <String>587</String>
                     <String>853</String>
                     <String>1068</String>
                     <String>2360</String>
                  </IdList>
               </DeleteUser>
            </TransactionOperation>
         </Transaction>
      </AdminOperation>
   </Administrate>
</SOAP-ENV:Body>
```

A request to delete a user or list of users returns an empty Administrate response if the request succeeds. If the request fails, an error message appears.

Updating an item

You can update information about a user, a security role, a file, a folder, a notification group, a channel, a job schedule, an Encyclopedia volume property, or a file type. Updating an item is a two-part process. First, you specify the update operation to apply. Then, you specify the item or items to update.

For most items, there are three types of Update operations:

- Update a single item, using Name or Id.
- Update a list of items, using NameList or IdList.
- Update all items that match a given condition using Search.

To update a file type, use only NameList or Name.

The item you update determines the type of update operations available. For example, to update a notification group, you can add and remove users by name or ID, and change the group name. To update a file, you can add or remove dependencies, set and grant permissions, and change autoarchive rules. To update a file type, you can specify a web icon and a Windows icon.

Updating a job schedule

To change a property of a job schedule, use UpdateJobSchedule and change SetAttributes, SetParameters, SetSchedules, or another element of the operation. The following request changes the run frequency, the job priority, and the run time. Because this job is a transaction, each element of the update must succeed for the update to succeed.

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
         <Transaction>
            <TransactionOperation>
               <UpdateJobSchedule>
                  <UpdateJobScheduleOperationGroup>
                     <UpdateJobScheduleOperation>
                        <SetAttributes>
                           <RunLatestVersion>true
                           </RunLatestVersion>
                           <InputFileName>
                              /National Data
                              /Nationalforecast.rptdesign
                           </InputFileName>
                           <Priority>800</Priority>
                        </SetAttributes>
                        <SetParameters>
                           <RetryOption>VolumeDefault</RetryOption>
                        </SetParameters>
                        <SetSchedules>
                           <TimeZoneName>PST</TimeZoneName>
                           <ScheduleDetails>
                              <JobScheduleDetail>
                                 <ScheduleType>Daily</ScheduleType>
                                    <Daily>
                                       <FrequencyInDays>1
                                       </FrequencyInDays>
                                       <OnceADay>07:00:00</OnceADay>
                                    </Daily>
                              </JobScheduleDetail>
                           </ScheduleDetails>
                        </SetSchedules>
                     </UpdateJobScheduleOperation>
```

```
                </UpdateJobScheduleOperationGroup>
                <Id>1</Id>
             </UpdateJobSchedule>
          </TransactionOperation>
       </Transaction>
    </AdminOperation>
  </Administrate>
 </SOAP-ENV:Body>
```

Updating a channel

To update a channel, use SetAttributes to change the channel's name and other properties. In SetPermissions, update the roles or users who can access this channel and the privileges of each role or user to the channel. Identify the channel to update using a name, an ID, or a list of names or IDs.

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
      <UpdateChannel>
         <UpdateChannelOperationGroup>
            <UpdateChannelOperation>
               <SetAttributes>
                  <Name>Regional Sales</Name>
               </SetAttributes>
               <SetPermissions>
                  <Permission>
                     <RoleName>Headquarters Staff - Accounting
                     </RoleName>
                     <AccessRight>R</AccessRight>
                  </Permission>
                  <Permission>
                     <RoleName>Regional Managers</RoleName>
                     <AccessRight>RW</AccessRight>
                  </Permission>
               </SetPermissions>
               <IdList>
                  <String>2</String>
                  <String>286</String>
                  <String>341</String>
            </UpdateChannelOperation>
         </UpdateChannelOperationGroup>
         </IdList>
         <IgnoreDup>false</IgnoreDup>
      </UpdateChannel>
   </Administrate>
</SOAP-ENV:Body>
```

Moving a file or folder

MoveFile moves a file or folder from the working directory to a location you specify using the Target element.

When you use MoveFile, you can indicate whether to create a new version of the file or folder if one with the same name already exists in the target location. If you create a new version, BIRT iHub overwrites the existing file with every update.

Using MaxVersions, you can specify the maximum number of versions to maintain on a BIRT iHub. For example, if you maintain four versions of a file, BIRT iHub deletes the earliest version when you add a fifth version. At this point, BIRT iHub can archive the file, depending on the autoarchive rules that apply to the file.

There are three ways to move a file or folder:

- Move a single file or folder using Name or Id.
- Move all files or folders that match a given condition using Search.

  When you use Search, you can use the optional LatestVersionOnly element to move only the latest version of the file matching the search criteria. Search also supports recursive MoveFile operations.
- Move a list of files or folders using NameList or IdList.

The following request uses Name to move Timeshares.rptdesign to the Inventory directory. It also specifies that the Encyclopedia volume keeps up to three versions of this file at a time.

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
         <MoveFile>
            <Target>/Inventory</Target>
            <Name>/Timeshares.rptdesign</Name>
            <MaxVersions>3</MaxVersions>
         </MoveFile>
      </AdminOperation>
   </Administrate>
</SOAP-ENV:Body>
```

Copying a file or folder

CopyFile copies a file or folder from the working directory to a target location. You can indicate whether to create a new version of the file or folder if one with the same name already exists in the target location.

Using MaxVersions, you can specify the maximum number of versions to maintain on BIRT iHub. For example, if you maintain four versions of a file, BIRT

iHub deletes the earliest version when you copy a fifth version to the Encyclopedia volume. At this point, BIRT iHub can also archive the file, depending on the file's autoarchive rules.

There are three ways to copy a file or folder:

- Copy all files or folders that match a given condition using Search.

  When you use Search, you can use the optional LatestVersionOnly parameter to specify that BIRT iHub copies only the latest version of the file that matches the search criteria. Search also supports recursive CopyFile operations.
- Copy a list of files or folders using NameList or IdList.
- Copy a single file or folder using Name or Id.

Specify the target directory using the Target element of CopyFile. The following example copies two files to two different directories:

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
         <Transaction>
            <TransactionOperation>
               <CopyFile>
                  <Target>/Headquarters</Target>
                  <Name>/EmployeeList.rptdesign</Name>
               </CopyFile>
               <CopyFile>
                  <Target>/SouthwestSales</Target>
                  <Name>/Prospects.rptdesign</Name>
               </CopyFile>
            </TransactionOperation>
         </Transaction>
      </AdminOperation>
   </Administrate>
</SOAP-ENV:Body>
```

About composite operations and transactions

Certain Administrate operations can be composite operations. That is, a single request can perform multiple administrative tasks, such as CreateGroup, DeleteUser, and CreateRole. In general, the operations you can group into a single message are those that create, update, or delete an item in an Encyclopedia volume.

A transaction is a composite operation identified in the Transaction element of the schema. If a failure occurs anywhere in the transaction sequence, all operations in the transaction fail.

The following example shows a composite request to add a user, Kevin Neery, to three groups. The groups are identified by their iHub-generated ID numbers in the IdList element.

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
         <UpdateGroup>
            <UpdateGroupOperationGroup>
               <UpdateGroupOperation>
                  <AssignedToUsersByName>
                     <String>Kevin Neery</String>
                  </AssignedToUsersByName>
               </UpdateGroupOperation>
            </UpdateGroupOperationGroup>
            <IdList>
                  <String>1024</String>
                  <String>3772</String>
                  <String>3781</String>
               </IdList>
         </UpdateGroup>
      </AdminOperation>
   </Administrate>
</SOAP-ENV:Body>
```

About sequences in composite Administrate operations

The operations in a composite Administrate operation execute sequentially. If the first two updates succeed but the update fails on the third object, BIRT iHub saves the updates to the first two objects, sends an error message for the third operation, and does not process subsequent updates.

For example, a request calls for adding two users to four groups. The request succeeds for the first user. The request fails to add the second user to the third group. BIRT iHub saves all successful operations and does not process requests that follow the failure.

Working with a transaction

In a composite Administrate operation, the failure of any single operation does not invalidate operations that complete successfully before the failure. To require that all operations succeed before any updates can occur, create a transaction.

In the following example, the two principal operations are CreateGroup and UpdateUser. UpdateUser consists of several additional operations, such as assigning security roles to users and adding those same users to two groups,

including the group that this operation creates. The following operation performs all UpdateUser operations on every user in the NameList element:

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
         <CreateGroup>
            <Group>
               <Name>Marketing Directors</Name>
               <Description>Direct reports to the Marketing VP
               </Description>
            <Group>
         </CreateGroup>
         <Transaction>
            <TransactionOperation>
               <UpdateUser>
                  <UpdateUserOperationGroup>
                     <UpdateUserOperation>
                        <AddToGroupsByName>
                           <String>Marketing Directors</String>
                           <String>All Employees</String>
                        </AddToGroupsByName>
                        <AddToRolesByName>
                           <String>Administrator</String>
                           <String>Management Staff</String>
                        </AddToRolesByName>
                     </UpdateUserOperation>
                  </UpdateUserOperationGroup>
                  <NameList>
                     <String>Claude Normand</String>
                     <String>Akiko Takagishi</String>
                     <String>Colleen O'Grady</String>
                  </NameList>
               </UpdateUser>
            </TransactionOperation>
         </Transaction>
      </AdminOperation>
   </Administrate>
</SOAP-ENV:Body>
```

As with other composite operations, CreateUser transactions execute sequentially. The difference is that any failure within a transaction causes the entire transaction to fail. For example, if AddToRolesByName succeeds for Claude Normand and Akiko Takagishi but fails for Colleen O'Grady, no users receive administrative privileges. All users' privileges revert to their previous settings and the entire AddToGroupsByName sequence fails. CreateGroup comes before the transaction. If CreateGroup succeeds, the operation adds the new

group to the database because CreateGroup completed before the failure. Operations that follow the failed transaction do not run.

About TransactionOperation and AdminOperation

Many programming languages are unable to store an array of objects of different data types. These languages cannot use the Administrate or Transaction elements alone. An Administrate operation that contains several different suboperations is likely to fail in these languages. A programmer working with one of these languages can perform multiple tasks in a single operation by using the AdminOperation and TransactionOperation elements.

An AdminOperation request represents a single unit of work within an Administrate operation. A TransactionOperation request represents a single unit of work within a Transaction. The external programming language treats each AdminOperation request or TransactionOperation request as one object. BIRT iHub processes each task within an AdminOperation request or TransactionOperation request as a single unit of work. Because AdminOperation is subordinate to Administrate, AdminOperation can contain any number of transactions. These transactions can contain any number of TransactionOperation requests.

In the following example, Administrate consists of two AdminOperation requests. The second AdminOperation request contains a single Transaction that consists of two TransactionOperation requests.

```
<SOAP-ENV:Body>
   <Administrate>
      <AdminOperation>
         <CreateGroup> … </CreateGroup>
      </AdminOperation>
      <AdminOperation>
         <Transaction>
            <TransactionOperation>
               <CreateUser>
                  <User>
                     <Name>jsheboah</Name>
                     <Password>grandee</Password>
                     <EmailAddress>jsheb@corporate.com
                     </EmailAddress>
                     <SendNoticeForSuccess>true
                     </SendNoticeForSuccess>
                     <SendNoticeForFailure>true
                     </SendNoticeForFailure>
                  </User>
                  <IgnoreDup>true</IgnoreDup>
               </CreateUser>
            </TransactionOperation>
```

```
            <TransactionOperation>
               <CopyFile>
                  <Target>/Headquarters</Target>
                  <Name>/EmployeeList.rptdesign</Name>
               </CopyFile>
            </TransactionOperation>
         </Transaction>
      </AdminOperation>
   </Administrate>
</SOAP-ENV:Body>
```

In this example, if the first AdminOperation request succeeds, BIRT iHub saves the data about the new group and proceeds to the next request, CreateUser. If CreateUser fails, processing stops and BIRT iHub does not process CopyFile.

Searching within an Encyclopedia volume

There are two categories of Encyclopedia volume search operations:

- Select operations, which retrieve items or lists of items and properties you specify in the request
- Get operations, which retrieve all available properties of an item

These operations apply only to the Encyclopedia volume you specify in the TargetVolume element of the SOAP header. Identify the target Encyclopedia volume using TargetVolume and indicate the operation to use. For a Select operation, list the properties to retrieve with the item or list.

FetchHandle is available when you use the Search element of a Get or Select request. If the result set is larger than the maximum FetchSize, BIRT iHub returns a FetchHandle with the response to keep the connection open for the next group of results.

Selecting an item in an Encyclopedia volume

The following example requests a list of users in the Encyclopedia volume Roland. The Encyclopedia volume name is in the TargetVolume element. For each user, this operation requests the ID, name, e-mail address, and home folder. It asks for 100 users at a time in the search result.

```
<SOAP-ENV:Header>
   <TargetVolume>ROLAND</TargetVolume>
   <AuthId>W4RhQBq0jidFqdi+o+r57uy</AuthId>
   <Locale>he_IL</Locale>
</SOAP-ENV:Header>
```

```
<SOAP-ENV:Body>
   <SelectUsers>
      <ResultDef>
         <String>Id</String>
         <String>Name</String>
         <String>EmailAddress</String>
         <String>Homefolder</String>
      </ResultDef>
      <Search>
         <FetchHandle/>
         <CountLimit>300</CountLimit>
         <FetchSize>100</FetchSize>
         <FetchDirection>true</FetchDirection>
      </Search>
   </SelectUsers>
   </SOAP-ENV:Body>
```

This request returns the specified properties for each user in the Encyclopedia volume and a total count of users.

```
<SOAP-ENV:Body>
   <SelectUsersResponse>
      <Users>
         <User>
            <Name>Samuel Stein</Name>
            <Id>188</Id>
            <EmailAddress>stein@cowin.com</EmailAddress>
            <HomeFolder>/Sales</HomeFolder>
         </User>
         <User>
            <Name>Jack Morris</Name>
            <Id>143</Id>
            <EmailAddress>morris@keller.com</EmailAddress>
            <HomeFolder>/Marketing</HomeFolder>
         </User>
         …
      </Users>
      <TotalCount>300</TotalCount>
   </SelectUsersResponse>
</SOAP-ENV:Body>
```

Selecting a job or job list

SelectJobs returns states and information for job instances. To retrieve job schedules, use SelectJobSchedules. SelectJobs retrieves information about a single job, a list of jobs, or jobs that match a certain condition. The ResultDef element specifies the job properties to retrieve, as shown in the following example:

```
<SOAP-ENV:Body>
   <SelectJobs>
      <ResultDef>
         <String>JobName</String>
         <String>JobId</String>
         <String>State</String>
      </ResultDef>
      <Search>
         <Condition>
            <Fileld>Tokyo Forecasts</Fileld>
         </Condition>
      </Search>
   </SelectJobs>
</SOAP-ENV:Body>
```

This request returns properties for the specified job and any previously generated instances of the job. A job's state can be Pending, Running, Succeeded, Failed, Cancelled, or Expired. The response also includes the total count of jobs that match the condition.

```
<SOAP-ENV:Body>
   <SelectJobsResponse>
      <Jobs>
         <JobProperties>
            <JobId>2</JobId>
            <JobName>Tokyo Forecasts</JobName>
            <State>Succeeded</State>
         </JobProperties>
         <JobProperties>
            <JobId>1</JobId>
            <JobName>Tokyo Forecasts</JobName>
            <State>Scheduled</State>
         </JobProperties>
      </Jobs>
      <TotalCount>2</TotalCount>
   </SelectJobsResponse>
</SOAP-ENV:Body>
```

To retrieve information regarding job schedules, use SelectJobSchedules.

Getting Encyclopedia volume, printer, and file type information

Use the following Get requests to retrieve information about Encyclopedia volumes, printers, user printer settings, and file type parameters:

- GetVolumeProperties retrieves information about a specific Encyclopedia volume on BIRT iHub.

- GetSystemPrinters retrieves information about printers for the Encyclopedia volume to which the user is logged in.
- GetUserPrinterOptions retrieves information about a given user's printer settings for printers on the current Encyclopedia volume.
- GetFileTypeParameterDefinitions retrieves parameters associated with a given file type on the current Encyclopedia volume.

Getting Encyclopedia volume properties

To display information about the current Encyclopedia volume, specify the Encyclopedia volume in the header. Then use GetVolumeProperties and ResultDef to request details about one or more of the following properties:

- VolumeProperties requests metadata about the Encyclopedia volume, including the Encyclopedia volume name, the default viewing preference for the Encyclopedia volume, the default printer name, and information about how often to retry jobs.
- OnlineBackupSchedule requests the schedule for backing up the Encyclopedia volume.
- TranslatedRoleNames requests the security roles in the Encyclopedia volume.
- ExternalUserPropertyNames requests user names from an external source.
- PrinterOptions requests printer settings for printers that the Encyclopedia volume accesses.
- AutoArchiveSchedule requests the default schedule for aging and archiving files on the Encyclopedia volume.
- ArchiveLibrary requests the name of any archive libraries set for the volume.

Requesting general Encyclopedia volume properties

The following example requests metadata about an Encyclopedia volume named voltaire:

```
<SOAP-ENV:Header>
   <TargetVolume>voltaire</TargetVolume>
   <AuthId>Q4yxAKHFJg9FY0JssYijJI5XvkJqDO</AuthId>
   <Locale>en_US</Locale>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <GetVolumeProperties>
      <ResultDef>
         <String>VolumeProperties</String>
      </ResultDef>
   </GetVolumeProperties>
</SOAP-ENV:Body>
```

The preceding request returns known properties of an Encyclopedia volume.

```
<SOAP-ENV:Body>
   <GetVolumePropertiesResponse>
      <VolumeProperties>
         <Name>voltaire</Name>
         <ActuateVersion>7 Development</ActuateVersion>
         <ActuateBuildNumber>DEV021230</ActuateBuildNumber>
         <SecurityIntegrationOption>0</SecurityIntegrationOption>
         <MaxJobRetryCount>0</MaxJobRetryCount>
         <JobRetryInterval>0</JobRetryInterval>
         <DefaultViewingPreference>DHTML
         </DefaultViewingPreference>
         <DHTMLPageCaching>false</DHTMLPageCaching>
         <OnlineBackupMode>false</OnlineBackupMode>
      </VolumeProperties>
   </GetVolumePropertiesResponse>
</SOAP-ENV:Body>
```

Getting BIRT iHub System printer information

The following operation requests all available information about a printer named MIRTH:

```
<SOAP-ENV:Header>
   <TargetVolume>radium</TargetVolume>
   <AuthId>E4yxAKHFJg9FY0JssYijJI5XvkJqDOs</AuthId>
   <Locale>en_US</Locale>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <GetSystemPrinters>
      <PrinterName>MIRTH</PrinterName>
   </GetSystemPrinters>
</SOAP-ENV:Body>
```

The preceding request returns all available information about the printer.

```
<SOAP-ENV:Body>
   <GetSystemPrintersResponse>
      <Printers>
         <Printer>
            <Name>MIRTH</Name>
            <Manufacturer>HP</Manufacturer>
            <Model>HP LaserJet 4050 Series PS</Model>
            <Location>Near Sales Area</Location>
            <Description>4050N</Description>
            <Orientation>PORTRAIT</Orientation>
            <PageSize>Letter</PageSize>
            <Scale>100</Scale>
```

```
            <Resolution>300 X 300</Resolution>
            <NumberOfCopies>1</NumberOfCopies>
            <Collation>false</Collation>
            <PaperTray>Automatically Select</PaperTray>
            <Duplex>SIMPLEX</Duplex>
            <ColorMode>true</ColorMode>
            <SupportOrientation>true</SupportOrientation>
            <OrientationOptions>
               <String>PORTRAIT</String>
               <String>LANDSCAPE</String>
               <String>AUTO</String>
            </OrientationOptions>
            <SupportPageSize>true</SupportPageSize>
            <PageSizeOptions>
               <String>Letter</String>
               <String>Letter Small</String>
               …
            </PageSizeOptions>
            <SupportScale>true</SupportScale>
            <ScaleOptions>
               <Integer>100</Integer>
            </ScaleOptions>
            <SupportResolution>true</SupportResolution>
            <ResolutionOptions>
               <String>300 X 300</String>
               <String>600 X 600</String>
            </ResolutionOptions>
            <SupportNumberOfCopies>false</SupportNumberOfCopies>
            <SupportCollation>true</SupportCollation>
            <SupportPaperTray>true</SupportPaperTray>
            <PaperTrayOptions>
               <String>Automatically Select</String>
               …
            </PaperTrayOptions>
            <SupportDuplex>true</SupportDuplex>
            <DuplexOptions >
               <String>SIMPLEX</String>
               …
            </DuplexOptions>
            <SupportColorMode>false</SupportColorMode>
            <ColorModeOptions>
               <String>true</String>
            </ColorModeOptions>
         </Printer>
      </Printers>
   </GetSystemPrintersResponse>
</SOAP-ENV:Body>
```

Retrieving a user's printer settings

Printer settings include the user's default printer, the orientation and default paper size on each printer, and printer resolution. To retrieve the printer settings for a user, use GetUserPrinterOptions and specify the user by name or ID.

```
<SOAP-ENV:Header>
   <TargetVolume>radium</TargetVolume>
      <AuthId>E4yxAKHFJg9FY0JssYijJI5XvkJqDOs</AuthId>
      <Locale>en_US</Locale>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <GetUserPrinterOptions>
      <UserId>515</UserId>
   </GetUserPrinterOptions>
</SOAP-ENV:Body>
```

The preceding request returns the settings for every printer available to the user.

```
<SOAP-ENV:Body>
   <GetUserPrinterOptionsResponse>
   <PrinterOptions>
      <PrinterOptions>
         <PrinterName>Corbet</PrinterName>
         <IsDefaultPrinter>false</IsDefaultPrinter>
         <Orientation>PORTRAIT</Orientation>
         <PageSize>Letter</PageSize>
         <Scale>100</Scale>
         <Resolution>300 X 300</Resolution>
         <NumberOfCopies>160</NumberOfCopies>
         …
      </PrinterOptions>
      <PrinterOptions>
         <PrinterName>Griswold</PrinterName>
         <IsDefaultPrinter>false</IsDefaultPrinter>
         <Orientation>PORTRAIT</Orientation>
         …
      </PrinterOptions>
   </PrinterOptions>
   </GetUserPrinterOptionsResponse>
</SOAP-ENV:Body>
```

Retrieving parameter definitions for a file type

GetFileTypeParameterDefinitions supports retrieving the parameter definitions for all files of a given type on an Encyclopedia volume.

```
<SOAP-ENV:Body>
   <GetFileTypeParameterDefinitions>
      <FileType>NewType</FileType>
   </GetFileTypeParameterDefinitions>
</SOAP-ENV:Body>
```

The ParameterDefinition element of the response returns details about every parameter associated with the specified file type. If the file type parameter is an ad hoc parameter, you must set ColumnName and ColumnType parameters in ParameterDefinition. The following example includes two parameter definitions:

```
<SOAP-ENV:Body>
   <GetFileTypeParameterDefinitionsResponse>
      <ParameterList>
         <ParameterDefinition>
            <Name>City</Name>
            <DataType>String</DataType>
            <DefaultValue>Boston</DefaultValue>
            <IsRequired>true</IsRequired>
            <IsPassword>false</IsPassword>
            <IsHidden>false</IsHidden>
            <IsAdHoc>false</IsAdHoc>
         </ParameterDefinition>
         <ParameterDefinition>
            <Name>Customer</Name>
            <DataType>String</DataType>
            <IsRequired>true</IsRequired>
            <IsPassword>false</IsPassword>
            <IsHidden>false</IsHidden>
            <DisplayName>Client</DisplayName>
            <IsAdHoc>false</IsAdHoc>
         </ParameterDefinition>
      </ParameterList>
   </GetFileTypeParameterDefinitionsResponse>
</SOAP-ENV:Body>
```

Extracting parameter definitions

Using the Actuate Information Delivery API, you can extract parameter definitions from a parameter values file. For example, you can determine whether a parameter is ad hoc, whether it is hidden, and what data type it is. To retrieve the parameter definitions from a file, use ExtractParameterDefinitionsFromFile and identify the file by name and file-type extension. Then submit the file as an attachment to the request or embed it in the request.

```
<SOAP-ENV:Body>
   <ExtractParameterDefinitionsFromFile>
      <Content>
         <ContentId>ROD_param.rop</ContentId>
         <ContentType>binary</ContentType>
         <ContentLength>62000</ContentLength>
      </Content>
   </ExtractParameterDefinitionsFromFile>
</SOAP-ENV:Body>
```

The preceding request returns a definition for each parameter in the attachment.

```
<SOAP-ENV:Body>
   <ExtractParameterDefinitionsFromFileResponse>
      <ParameterDefinitions>
         <ParameterDefinition>
            <Name>DCP_DEBUG_LEVEL</Name>
            <DataType>Integer</DataType>
            <DefaultValue>100</DefaultValue>
            <IsRequired>false</IsRequired>
            <IsPassword>false</IsPassword>
            <IsHidden>false</IsHidden>
            <IsAdHoc>false</IsAdHoc>
         </ParameterDefinition>
         <ParameterDefinition>
            <Name>DCP_ESPRESSO_LIB</Name>
            <DataType>String</DataType>
            <IsRequired>false</IsRequired>
            <IsPassword>false</IsPassword>
            <IsHidden>false</IsHidden>
            <IsAdHoc>false</IsAdHoc>
         </ParameterDefinition>
         …
      </ParameterDefinitions>
   </ExtractParameterDefinitionsFromFileResponse>
</SOAP-ENV:Body>
```

Exporting file parameters

ExportParameterDefinitionsToFile converts parameter definitions to an attached file that the client application can use as a report's parameter values file. Because the parameter in the following request is an ad hoc parameter, you must use ColumnName and ColumnType. ColumnName is the name of the database column to which this ad hoc parameter applies. ColumnType is the Actuate data type of the column.

```
<SOAP-ENV:Body>
   <ExportParameterDefinitionsToFile>
      <ParameterList>
         <ParameterDefinition>
            <Group>Clients</Group>
            <Name>OPS_HOME</Name>
            <DataType>String</DataType>
            <DefaultValue>default</DefaultValue>
            <IsRequired>True</IsRequired>
            <IsPassword>False</IsPassword>
            <IsHidden>True</IsHidden>
            <DisplayName>Home</DisplayName>
            <IsAdHoc>True</IsAdHoc>
            <ColumnName>Operations</ColumnName>
            <ColumnType>String</ColumnType>
         </ParameterDefinition>
      </ParameterList>
   </ExportParameterDefinitionsToFile>
</SOAP-ENV:Body>
```

This request returns the attachment and provides descriptive data about the attachment in the Content element.

```
<SOAP-ENV:Body>
   <ExportParameterDefinitionsToFileResponse>
      <Content>
         <ContentId>ExportParameterDefinitionsToFile</ContentId>
         <ContentType>application/octet-stream
         </ContentType>
      </Content>
   </ExportParameterDefinitionsToFileResponse>
</SOAP-ENV:Body>
```

Executing a predefined Encyclopedia volume command

Using the Actuate Information Delivery API, you can execute one of the following predefined Encyclopedia volume commands shown in Table 5-2.

Table 5-2 Predefined Encyclopedia volume commands

Command	Description
StartArchive	Starts the archive process for the Encyclopedia volume
StartPartitionPhaseOut	Starts the process of moving data out of a partition

The following request starts partition phaseout in 60 seconds:

```
<SOAP-ENV:Body>
   <ExecuteVolumeCommand>
      <VolumeName>Autry</VolumeName>
      <Command>StartPartitionPhaseOut</Command>
      <GracePeriodInSeconds>60</GracePeriodInSeconds>
   </ExecuteVolumeCommand>
</SOAP-ENV:Body>
```

This request returns a status for the command, either Succeeded or Failed.

```
<SOAP-ENV:Body>
   <ExecuteVolumeCommandResponse>
      <Status>Succeeded</Status>
   </ExecuteVolumeCommandResponse>
</SOAP-ENV:Body>
```

Diagnosing reporting environment problems

Use the Ping request to test whether a specific component of the reporting environment is operational and to retrieve other information. A Ping request must specify a destination. Using the Information Delivery API, you can test the following destinations:

- The Message Distribution service (MDS)
- A BIRT iHub node running the Encyclopedia engine (EE)
- A BIRT iHub node running the Factory service (FS)
- A BIRT iHub node running the View service (VS)
- An Actuate open server driver (OSD)
- A connection to a data source

If a destination is not operational, BIRT iHub returns an error message. If a destination is operational, the response depends on the Ping request you send. For example, you can request a simple timestamp that shows the elapsed time between when a component receives the request and when it sends a reply. You also can request more detailed information.

A Ping request to the MDS has no security restrictions. For all other components, the request is subject to Encyclopedia volume authentication. The user must be an Encyclopedia volume administrator or a user in the Operator security role.

About Ping request options

A Ping request must specify a destination to test, expressed as a string. It can also specify an action to take, such as reading a file or writing a temporary file,

and the level of detail to include in the response. Table 5-3 describes the elements of a Ping request.

Table 5-3 Ping request elements

Element	Description
Destination	The destination to test. This element is required. Valid values are: ■ MDS (Message Distribution service) ■ EE (Encyclopedia engine) ■ FS (Factory service) ■ VS (View service) ■ OSD (open server driver)
Action	An optional element specifying the action to take at the destination. Valid values are: ■ Echo—Echoes data specified in the Payload parameter. ■ ReadFile—Opens a specified Encyclopedia volume file, reads its content, and closes the file. Destination must be EE, FS, or VS. ■ WriteFile—Creates a temporary file in a partition, writes a specified number of bytes, closes the file, and deletes it. Destination must be EE or FS. ■ Connect—Connects to a data source. If you do not specify a value, the destination component responds to the request without taking another action.
Mode	An optional element specifying the level of detail in the Ping response. Valid values are: ■ Concise—Returns the elapsed time between a component's receipt of the request and the time the component sends a reply. ■ Normal—Returns the names of components in the test path and the timestamps of the request entering and leaving each component ■ Trace—Returns the timestamp of the request entering and leaving major subcomponents of the component being tested. For example, a request to a node running the Encyclopedia service can provide a timestamp for when the request enters and leaves the process queue. A Ping request in Trace mode also can return diagnostic information other than timing. For example, a request to test writing a temporary file to a partition can return the amount of free disk space on the partition.

Table 5-3 Ping request elements

Element	Description
Server	An optional element specifying which instance of a Factory service or View service to test. Works with the ProcessID parameter. To test all available instances of the Factory or View service, use an asterisk (*). If you do not use Server, the iHub load balancing mechanism allocates an available instance of the requested service to respond to the Ping request.
ProcessID	With Server, this optional element specifies the process ID of the Factory or View service to test.
FileName	If the Action is ReadFile, this element is required to indicate the Encyclopedia volume file to read. If you ping an open server driver, FileName specifies the executable file to prepare for execution.
ConnectionProperties	If the Action is Connect, ConnectionProperties specifies properties required to connect to a data source, such as user name and password. ConnectionProperties must define the DBType. Valid values are: ■ DB2 ■ Informix ■ MSSQL ■ ODBC ■ Oracle ■ Progress ■ Progress SQL92 ■ Sybase A database can require the Ping request to define additional properties.
Payload	A Ping request can include binary data that returns in the response. This binary data is called the payload. If the Action is Echo, you can specify the length of the payload data.
PartitionName	If the Action is WriteFile, this optional element specifies the name of the partition on which to create the temporary file.
NumBytes	If the Action is ReadFile or WriteFile, specifies the number of bytes to read or write. If you do not specify NumBytes or the value is 0, the Factory uses the default value of 10 KB.

Sending a Ping request in Concise mode

The following example requests an echo from the Encyclopedia service and asks for the response in Concise mode. The request includes binary payload data for the response to return.

```
<SOAP-ENV:Header>
   <AuthId>MvEQLSkwhYEHEBl8PqbCum5+MLDk=</AuthId>
   <Locale>en_US</Locale>
   <TargetVolume>Legion</TargetVolume>
   <DelayFlush>true</DelayFlush>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <Ping>
      <Destination>EE</Destination>
      <Action>Echo</Action>
      <Mode>Concise</Mode>
      <Payload>******************************</Payload>
   </Ping>
</SOAP-ENV:Body>
```

A Ping response in Concise mode sends the total elapsed time to send the request and receive the response. If the request includes payload data, the data returns in the response.

```
<SOAP-ENV:Body>
   <PingResponse>
      <Reply>Ping reply from EE received. Time elapsed= 10 ms
         </Reply>
      <Payload>******************************</Payload>
   </PingResponse>
</SOAP-ENV:Body>
```

Sending a Ping request in Normal mode

The following request asks the Encyclopedia service to read a file in the target volume and send a response in Normal mode:

```
<SOAP-ENV:Header>
   <AuthId>84yxAKHFJ=</AuthId>
   <Locale>en_US</Locale>
   <TargetVolume>Legion</TargetVolume>
   <DelayFlush>true</DelayFlush>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <Ping>
      <Destination>EE</Destination>
      <Action>ReadFile</Action>
```

```
      <Mode>Normal</Mode>
      <FileName>WesternRegionalShipping.rptdesign</FileName>
   </Ping>
</SOAP-ENV:Body>
```

A response in Normal mode shows each major component in the test path and the time each component receives the request and sends a reply. The following response shows request and reply times for the MDS and Encyclopedia service:

```
<SOAP-ENV:Body>
   <PingResponse>
      <Reply>
         MDS(Legion): Elapsed= 0 ms
         Received: 09:31:17.304
         Reply: 09:31:17.304
         EncycEngine(Legion): Elapsed= 0 ms
         Received: 09:31:17.304
         Reply: 09:31:17.304
         Action: Read 10240 bytes from file
      </Reply>
   </PingResponse>
</SOAP-ENV:Body>
```

Sending a Ping request in Trace mode

The following request asks the Encyclopedia service to read a file in the target volume and send a response in Trace mode:

```
<SOAP-ENV:Header>
   <AuthId>84yxAKHFJg9FY08PqbCum5+MLDk=</AuthId>
   <Locale>en_US</Locale>
   <TargetVolume>Legion</TargetVolume>
   <DelayFlush>true</DelayFlush>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <Ping>
      <Destination>EE</Destination>
      <Action>ReadFile</Action>
      <Mode>Trace</Mode>
      <FileName>WesternRegionalShipping.rptdesign</FileName>
   </Ping>
</SOAP-ENV:Body>
```

A response in Trace mode provides more detailed information about the objects in the test path than a response in Normal mode.

```
<SOAP-ENV:Body>
   <PingResponse>
      <Reply>
         09:31:37.963: MDS(Legion) received Ping message
         09:31:37.963: MDS(Legion) forwarding Ping request to
            node end00166
         09:31:37.963: EncycEngine(Legion) received Ping message
         09:31:37.963: EncycEngine(Legion) found file
            &apos;Design1.rptdesign&apos;. Time= 0 ms
         09:31:37.973: EncycEngine(Legion) opened file in 10 ms
         09:31:37.973: EncycEngine(Legion) read 10240 bytes from
            file in 0 ms
         09:31:37.973: EncycEngine(Legion) replying to Ping
            message. Elapsed= 10 ms
         09:31:37.963: MDS(Legion) received Ping reply from node
            end00166. Roundtrip= 10 ms
         09:31:37.973: MDS(Legion) replying to Ping message.
            Elapsed= 10 ms
      </Reply>
   </PingResponse>
</SOAP-ENV:Body>
```

Monitoring BIRT iHub information

A system administrator can retrieve data about BIRT iHub. The Actuate Information Delivery API provides operations that support:

- Getting information about BIRT iHub
- Getting information about a running or pending job
- Getting information about Factory service processes

These operations help determine whether to cancel a report that blocks or overloads the system. To monitor BIRT iHub System information, you must be an iHub System administrator and use SystemLogin to get an administrator's AuthId.

Getting information about BIRT iHub

To obtain a list of BIRT iHub nodes and their properties, send GetSystemServerList. There are no request parameters for this operation. Identify the target volume in the SOAP header, as shown in the following example:

```
<SOAP-ENV:Header>
   <TargetVolume>RADIUM</TargetVolume>
   <AuthId>+zxJgKuMBr4lC1psWCGajW8GXIp7</AuthId>
   <Locale>en_US</Locale>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <GetSystemServerList>
   </GetSystemServerList>
</SOAP-ENV:Body>
```

The response includes the list of BIRT iHub nodes, their state information, a list of services running on each node, and whether the node is the cluster master. ServerVersion is the Actuate release number of the node running on the machine.

```
<SOAP-ENV:Body>
   <GetSystemServerListResponse>
      <ServerList>
         <ServerInformation>
            <ServerName>Homeland</ServerName>
            <ServerStatusInformation>
               <ServerState>ONLINE</ServerState>
            </ServerStatusInformation>
            <ServiceList>
               <Service>Request</Service>
               <Service>Viewing</Service>
               <Service>Generation</Service>
            </ServiceList>
            <OwnsVolume>true</OwnsVolume>
            <ServerVersionInformation>
               <ServerVersion>11</ServerVersion>
               <ServerBuild>Production</ServerBuild>
               <OSVersion>Windows 7 Service Pack 1
               </OSVersion>
            </ServerVersionInformation>
            <ChangesPending>false</ChangesPending>
         </ServerInformation>
         …
      </ServerList>
   </GetSystemServerListResponse>
</SOAP-ENV:Body>
```

Getting information about a running or pending job

GetFactoryServiceJobs returns a list of synchronous jobs that are either running or pending on the node specified by the TargetServer element in the SOAP header. You can also request a list of synchronous and asynchronous reports currently running on the target machine and specific job properties.

GetFactoryServiceJobs always returns the following data in Table 5-4 for a running or a pending job.

Table 5-4 Job data for GetFactoryServiceJobs

Pending synchronous job	Running job
ConnectionHandle	ConnectionHandle for a synchronous job
IsTransient	IsSyncJob
ObjectId	ObjectId for a synchronous job
Volume	IsTransient for a synchronous job
	JobId for an asynchronous job
	Volume for synchronous and asynchronous jobs

You can use PendingSyncJobsResultDef and RunningJobsResultDef to define other properties to retrieve for a pending or a running job, as shown in the following example:

```
<SOAP-ENV:Header>
   <AuthId>G4RhQBxOo0HDEQLSkwhYEHFr2ZcApO1AA==</AuthId>
   <TargetServer>Tokyo</TargetServer>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <GetFactoryServiceJobs>
      <PendingSyncJobsResultDef>
         <String>Owner</String>
         <String>ExecutableFileName</String>
         <String>ExecutableVersionNumber</String>
         <String>PendingTime</String>
         <String>ResourceGroup</String>
      </PendingSyncJobsResultDef>
      <RunningJobsResultDef>
         <String>Owner</String>
         <String>ExecutableFileName</String>
         <String>StartTime</String>
         <String>RunningTime</String>
         <String>ExecutionTimeout</String>
         <String>ResourceGroup</String
      </RunningJobsResultDef>
   </GetFactoryServiceJobs >
</SOAP-ENV:Body>
```

This request returns the default properties and those requested in ResultDef. For a pending job, the response includes details such as the full path to the executable file, the version number, the resource group to which the job is assigned, and the time since the report entered the queue, expressed in seconds. For a running job, the response can include the number of seconds the report has been running and

the number of seconds remaining before job execution times out. The following example shows a response that includes synchronous and asynchronous jobs:

```
<SOAP-ENV:Body>
   <GetFactoryServiceJobsResponse>
      <PendingSyncJobs>
         <PendingSyncJob>
            <ConnectionHandle>1DelRhsJO8Jo</ConnectionHandle>
            <ObjectId>25</ObjectId>
            <IsTransient>true</IsTransient>
            <Volume>Dresden</Volume>
            <Owner>Ray Morrell</Owner>
            <ExecutableFileName>/Marketing/Campaign2008.rptdesign
            </ExecutableFileName>
            <ExecutableVersionNumber>2</ExecutableVersionNumber >
            <PendingTime>921</PendingTime >
            <ResourceGroup>Default Sync</ResourceGroup>
         </PendingSyncJob>
         …
      </PendingSyncJobs>
      <RunningJobs>
         <RunningJob>
            <IsSyncJob>true</IsSyncJob>
            <ConnectionHandle>1DelRhsJO8Jo</ConnectionHandle>
            <ObjectId>125</ObjectId>
            <IsTransient>true</IsTransient>
            <Volume>Corinth</Volume>
            <Owner>Pablo Ruiz</Owner>
            <ExecutableFileName>/Forecasts/detail.rptdesign
            </ExecutableFileName>
            <StartTime>2008-10-03 06:11:51</StartTime>
            <RunningTime>821</RunningTime>
            <ExecutionTimeout>79</ExecutionTimeout>
            <ResourceGroup>Default Sync</ResourceGroup>
         </RunningJob>
         <RunningJob>
            <IsSyncJob>false</IsSyncJob>
            <ConnectionHandle>1DelRhsJO8Jo</ConnectionHandle>
            <ObjectId>3031</ObjectId>
            <IsTransient>true</IsTransient>
            <Volume>Rubio</Volume>
            <Owner>Frank Kitada</Owner>
            <ExecutableFileName>/Forecasts/regions.rptdesign
            </ExecutableFileName>
            <StartTime>2008-10-03 06:13:41</StartTime>
            <RunningTime>438</RunningTime>
```

```
            <ExecutionTimeout>500</ExecutionTimeout>
            <ResourceGroup>Default Async</ResourceGroup>
         </RunningJob>
      </RunningJobs>
   </GetFactoryServiceJobs Response>
</SOAP-ENV:Body>
```

Getting information about Factory service processes

GetFactoryServiceInfo provides data about Factory processes currently running on a specific BIRT iHub. This operation supports checking the number of running and pending jobs against the capacity of BIRT iHub. It shows the percent of disk space in use, how many synchronous jobs are pending and running, the cache size, and other information useful to an administrator.

To use this operation, identify the target BIRT iHub in the SOAP envelope header and send GetFactoryServiceInfo without request parameters.

```
<SOAP-ENV:Header>
   <AuthId>G4RhQBqSkwhYEHFr2ZcApO1AA==</AuthId>
   <TargetServer>Wizard</TargetServer>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <GetFactoryServiceInfo></GetFactoryServiceInfo>
</SOAP-ENV:Body>
```

The response to this request indicates that there are five synchronous jobs pending on a BIRT iHub named Wizard, which can queue a maximum of 100 synchronous jobs. Four jobs are running, three of which are synchronous.

The response also shows:

- SyncJobQueueWait, the length of time before the system deletes a synchronous job from the queue, expressed in seconds.
- TransientReportTimeout, the maximum length of time before the system deletes a temporary report from the synchronous cache, expressed in minutes. The configuration file sets this value.
- CurrentTransientReportTimeout, the actual length of time, expressed in minutes, before the system deletes a temporary report from the synchronous cache. BIRT iHub sets this value internally at run time. This value must be less than the TransientReportTimeout value.
- MaxSyncJobRuntime, the maximum job execution time, expressed in seconds.

The response expresses cache sizes in megabytes. GetFactoryServiceInfo always returns all the data shown in the following example:

```
<SOAP-ENV:Body>
   <GetFactoryServiceInfoResponse>
      <ServerName>Wizard</ServerName>
      <PendingSyncJobs>5</PendingSyncJobs>
      <SyncJobQueueSize>100</SyncJobQueueSize>
      <RunningSyncJobs>3</RunningSyncJobs>
      <RunningJobs>4</RunningJobs>
      <SyncFactoryProcesses>3</SyncFactoryProcesses>
      <MaxFactoryProcesses>4</MaxFactoryProcesses>
      <TransientReportCacheSize>500</TransientReportCacheSize>
      <PercentTransientReportCacheInUse>7
      </PercentTransientReportCacheInUse>
      <CurrentTransientReportTimeout>900
      </CurrentTransientReportTimeout>
      <TransientReportTimeout>1800</TransientReportTimeout>
      <SyncJobQueueWait>600</SyncJobQueueWait>
      <MaxSyncJobRuntime>900</MaxSyncJobRuntime>
   <GetFactoryServiceInfoResponse>
</SOAP-ENV:Body>
```

Monitoring or cancelling a request for a synchronous report

The Actuate Information Delivery API supports monitoring the progress of a synchronous report. If report generation takes longer than a configurable period of time, an Encyclopedia volume administrator can monitor the progress of the report and determine whether to cancel the request.

To monitor any system event, you must first log in as Administrator, using the SystemLogin operation. This section explains how to monitor and cancel a request for a synchronous report using the Actuate Information Delivery API.

Monitoring a request for a synchronous report

GetSyncJobInfo retrieves information about synchronous jobs on BIRT iHub. For example, you can get the status of the report, its position in the queue, the name of the BIRT iHub machine on which it is pending, whether the report is transient or persistent, and how soon the request times out. The status of a synchronous job is Completed, Pending, Running, or Failed. GetSyncJobInfo also returns an error description if the status is Failed.

To submit a GetSyncJobInfo request, you need a ConnectionHandle and the ObjectId of the requested job. ConnectionHandle returns in the response to ExecuteReport.

```
<SOAP-ENV:Header>
   <AuthId>G4RhQBq0jidFqdi+o+Kh5JDhWA==</AuthId>
   <ConnectionHandle>QBq0jidFqd</ConnectionHandle>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <GetSyncJobInfo>
      <ObjectId>356</ObjectId>
   </GetSyncJobInfo>
</SOAP-ENV:Body>
```

This request returns known details about the job. In the following response, PendingTime is the number of seconds since the job entered the queue. QueueTimeout is the number of seconds remaining before BIRT iHub deletes the job from the queue. The default value for QueueTimeout is 600 seconds. An Encyclopedia volume administrator can configure a different value, to a maximum of 999 seconds. The response also shows the path to the executable file that creates the output, whether the report is transient, the resource group to which the job is assigned, and other information about the job.

```
<SOAP-ENV:Body>
   <GetSyncJobInfoResponse>
      <Status>Pending</Status>
      <PendingSyncJob>
         <ConnectionHandle>HxTYGwG77</ConnectionHandle>
         <ObjectId>356</ObjectId>
         <IsTransient>true</IsTransient>
         <Volume>Monaco</Volume>
         <ServerName>Melville</ServerName>
         <Owner>Bob Carlton</Owner>
         <ExecutableFileName>/Forecasts/Detail.rptdesign
         </ExecutableFileName>
         <ExecutableVersionNumber>13</ExecutableVersionNumber>
         <ResourceGroup>Default Sync</ResourceGroup>
         <SubmissionTime>2008-09-11 09:30:47</SubmissionTime>
         <PendingTime>124</PendingTime>
         <QueueTimeout>176</QueueTimeout>
         <QueuePosition>26</QueuePosition>
      </PendingSyncJob>
   </GetSyncJobInfoResponse>
</SOAP-ENV:Body>
```

Cancelling a request for a synchronous report

CancelReport supports cancelling a request for a synchronous report. Any user who can send a request for a report can cancel their own request. Only a system administrator can cancel the request of another user.

CancelReport returns one of the following status messages:

- Failed, meaning that the cancellation request failed because of authentication issues or another cause
- Succeeded, meaning that the request is cancelled
- InActive, meaning that report generation was complete when BIRT iHub received the request

A report user can cancel a request for a synchronous report at two stages:

- After submitting the synchronous job request and receiving a ConnectionHandle
- After receiving the first page of a progressive report

The report user cannot cancel a request until the response to ExecuteReport returns a ConnectionHandle.

The following example shows how to cancel a request for synchronous report generation using the ConnectionHandle from the ExecuteReport request and identifying the report to cancel by ObjectId. ObjectId comes from the response to ExecuteReport. AuthId must the AuthId of the user who generated the report or the system administrator.

```
<SOAP-ENV:Header>
   <AuthId>Ft7m4truCmY7k5EQLSkwhYEHEfic1c6pcWhvcxA==</AuthId>
   <Locale>en_us</Locale>
   <ConnectionHandle>RYEMWxKxEsxq0mQCiXCZHB8NiLDFwq1Q=
   </ConnectionHandle>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <CancelReport>
      <ObjectId>435</ObjectId>
   </CancelReport>
</SOAP-ENV:Body>
```

The preceding request returns the status of the cancellation.

```
<SOAP-ENV:Body>
   <CancelReportResponse>
      <Status>Succeeded</Status>
   <CancelReportResponse>
</SOAP-ENV:Body>
```

Part Two

Developing Actuate Information Delivery API applications

Chapter

6

Developing Actuate Information Delivery API applications using Java

This chapter consists of the following topics:

- About the Apache Axis 1.4 client
- About the Actuate Information Delivery API framework
- Developing Actuate Information Delivery API applications
- SOAP-based event web service operations and data types

About the Apache Axis 1.4 client

BIRT iHub System contains a WSDL document that defines an Actuate web services schema for the Apache Axis 1.4 client. The Apache Axis 1.4 client is a Java-based framework for constructing a SOAP processor.

The Actuate Information Delivery API framework uses elements of the Apache Axis 1.4 code libraries to support the following features:

- The code emitter, org.apache.axis.wsdl.WSDL2Java, generates the Java source code package, com.actuate.schemas, from the Actuate WSDL document.

  The package contains the classes, including proxies, that you can use to write an Actuate Information Delivery API application to communicate with BIRT iHub System using SOAP messaging.
- The SOAP processor provides automatic JavaBean serialization and deserialization, using the com.actuate.schemas proxies, to encode and decode SOAP messages.

The following sections describe how to generate the com.actuate.schemas library and list the third-party code libraries required by the Actuate Information Delivery API development environment.

Generating the com.actuate.schemas library

The Apache Axis 1.4 client ships with BIRT iHub Integration Technology. In the web services examples, the Apache Axis 1.4 client is in the following directory:

```
\Actuate\ServerIntTech\Web Services\Examples\Axis Client
```

You can generate the source code for the package, com.actuate.schemas, compile the classes, and archive the classes into a library file, using one of the following supplied methods:

- Batch

  To use the batch file, build.bat, open a command prompt. Navigate to the Axis Client directory. At the command line, type:

  ```
  build
  ```
- Apache Ant
  - To use Apache Ant, you must first install the build tool on your computer. To obtain the software and installation instructions, go to the Apache Ant Project web site at http://ant.apache.org/.
  - To use Apache Ant, open a command prompt. Navigate to the Axis Client directory.

- To generate the source code for the package, com.actuate.schemas, type:

  ```
  ant
  ```

- To compile the source code and generate the com.actuate.schemas library, type:

  ```
  ant dist
  ```

- To generate Javadoc for the com.actuate.schemas library, type:

  ```
  ant documentation
  ```

Each of these methods performs the operations by setting the properties that specify the locations and file names for the following resources:

- WSDL document
  The Actuate WSDL document is available at the following URL:

  ```
  http://localhost:8000/wsdl/v11/axis/all
  ```

- Source code
  The code emitter, WSDL2Java, generates Java source code from the Actuate WSDL document, placing the package, com.actuate.schemas, in the directory, \Actuate\ServerIntTech\Web Services\Examples\Axis Client\source.

- Compiled code
  Both methods use javac to compile the source code, placing the compiled package, com.actuate.schemas, in the directory, \Actuate \ServerIntTech\Web Services\Examples\Axis Client\build.

- Library files
  Both methods use jar to archive the package, com.actuate.schemas. The batch method places the library file, idapi.jar, in the directory, \Actuate \ServerIntTech\Web Services\Examples\Axis Client\lib.

  The Apache Ant method places the library file, ActuateClient-${DSTAMP}.jar, in the directory, \Actuate\ServerIntTech\Web Services\Examples \Axis Client\dist\lib. DSTAMP is a variable in the file, build.xml, that represents the time when the JAR file was created. To run the example applications, copy the file, ActuateClient-${DSTAMP}.jar, to the directory, \Actuate\ServerIntTech\Web Services\Examples\Axis Client \dist\lib, and change the file name to idapi.jar.

About third-party code libraries

The BIRT iHub Integration Technology example applications require code libraries from the following third-party sources:

- Apache Axis

  ```
  http://xml.apache.org/axis
  ```

- Apache Log4j

  `http://logging.apache.org`

- Jakarta Commons

  `http://jakarta.apache.org/commons`

- JavaBeans™ Activation Framework (JAF)

  `http://java.sun.com/products/javabeans/glasgow/jaf.html`

- JavaMail™

  `http://java.sun.com/products/javamail`

- SOAP with Attachments API for Java™

  `http://java.sun.com/xml/downloads/saaj.html`

- SourceForge.net Web Services Description Language for Java Toolkit (WSDL4J)

  `http://sourceforge.net/projects/wsdl4j`

- Xerces XML Parser

  `http://xml.apache.org/xerces2-j`

Actuate supplies the necessary libraries in the \lib directory of the BIRT iHub Integration Technology installation.

About the Actuate Information Delivery API framework

The org.apache.axis.wsdl.WSDL2Java tool generates the Actuate Information Delivery API application framework based on the Actuate WSDL document definitions. This framework contains the client-side bindings that the Actuate IDAPI application requires to implement SOAP processing.

The SOAP processor serializes, or transforms, a remote procedure call by the application into an XML-based SOAP message that asks BIRT iHub to perform a web service. The application sends the request across the network using the HyperText Transfer Protocol (HTTP) transport layer.

BIRT iHub receives the request and deserializes the SOAP message. BIRT iHub performs an appropriate action and sends a response, in the form of a SOAP message, back to the application. The SOAP processor embedded in the Actuate Information Delivery API framework automates the serialization and deserialization of JavaBeans, relieving the developer of the necessity to program the application at this level. The framework code is visible in the com.actuate.schemas classes.

The following sections describe these key elements of the framework as background information to provide the developer with an understanding of the way the Actuate Information Delivery API framework operates.

Using a data type from a WSDL document to generate a JavaBean

When you generate the Actuate Information Delivery API source code, the WSDL2Java tool builds a Java class from each WSDL type definition. For example, WSDL2Java translates the following Login type definition into its Java equivalent:

```
<xsd:complexType name="Login">
   <xsd:sequence>
      <xsd:element name="User" type="xsd:string"/>
      <xsd:element name="Password" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="EncryptedPwd" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Credentials" type="xsd:base64Binary"
         minOccurs="0"/>
      <xsd:element name="Domain" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="UserSetting" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="ValidateRoles" type="typens:
         ArrayOfString" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

The WSDL2Java tool gives the generated Java class the name that appears in the WSDL type definition. The class defines the attributes and data types for each WSDL element with corresponding accessor methods, as shown in the following code:

```
package com.actuate.schemas;

public class Login  implements java.io.Serializable {
   private java.lang.String user;
   private java.lang.String password;
   private java.lang.String encryptedPwd;
   private byte[ ] credentials;
   private java.lang.String domain;
   private java.lang.Boolean userSetting;
   private com.actuate.schemas.ArrayOfString validateRoles;
   public Login( ) {
   }
   …
```

```
   public java.lang.String getUser( ) {
      return user;
   }

   public void setUser(java.lang.String user) {
      this.user = user;
…
```

Using metadata to map XML to a Java type

Mapping XML to a Java type requires creating a collection of descriptors in the class to associate each Java attribute with its corresponding XML element. This mapping system manages any naming differences between the Java and XML pairs to support the serialization and deserialization of the data.

The WSDL2Java tool generates a static type descriptor for each Java and XML pair. The following code maps the qualified names of the Java attribute and XML element for User in the Login class:

```
…
private static org.apache.axis.description.TypeDesc typeDesc =
   new org.apache.axis.description.TypeDesc(Login.class, true);

   static {
      typeDesc.setXmlType(new javax.xml.namespace.QName(
         "http://schemas.actuate.com/actuate11", "Login"));
      org.apache.axis.description.ElementDesc elemField =
         new org.apache.axis.description.ElementDesc( );
      elemField.setFieldName("user");
      elemField.setXmlName(new javax.xml.namespace.QName(
         "http://schemas.actuate.com/actuate11", "User"));
      elemField.setXmlType(new javax.xml.namespace.QName(
         "http://www.w3.org/2001/XMLSchema", "string"));
      elemField.setNillable(false);
   typeDesc.addFieldDesc(elemField);
…
```

A class generated from a WSDL type is typically a JavaBean. The JavaBean uses classes from the org.apache.axis.encoding.ser package to encode and decode SOAP messages. In the following code example, getSerializer( ) instantiates and returns a reference to a BeanSerializer object using the Java and XML type descriptors:

```
…
public static org.apache.axis.encoding.Serializer getSerializer(
      java.lang.String mechType,
      java.lang.Class _javaType,
      javax.xml.namespace.QName _xmlType) {
```

```
   return
      new  org.apache.axis.encoding.ser.BeanSerializer(
         _javaType, _xmlType, typeDesc);
   }
…
```

Mapping the portType to a Service Definition Interface

WSDL2Java uses the portType and binding in the WSDL document to create the Service Definition Interface (SDI). The Service Definition Interface specifies the input and output messages of the request-response pairs for an operation and the service name and port.

The following WSDL code shows the specification of the input and output messages, Login and LoginResponse, and the binding of this request-response pair to the login operation:

```
…
<portType name="ActuateSoapPort">
   <operation name="login">
      <input message="wsdlns:Login"/>
      <output message="wsdlns:LoginResponse"/>
   </operation>
</portType>
…
<binding name="ActuateSoapBinding" type="wsdlns:ActuateSoapPort">
   <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="login">
      <soap:operation soapAction=""/>
      <input>
         <soap:body use="literal" parts="Request"/>
      </input>
      <output>
         <soap:body use="literal" parts="Response"/>
      </output>
   </operation>
</binding>
…
```

The following WSDL code shows the specification of the service and port names, and the binding of the port to a machine address:

```
…
<service name="ActuateAPI">
```

```
   <port name="ActuateSoapPort"
   binding="wsdlns:ActuateSoapBinding">
      <soap:address location="http://localhost:8000"/>
   </port>
</service>
…
```

An application uses this information to construct an interface to access the operations available from the service using a remote procedure call (RPC), as shown in the following code:

```
package com.actuate.schemas;

public interface ActuateSoapPort_PortType extends java.rmi.Remote
   {
   public com.actuate.schemas.LoginResponse
      login(com.actuate.schemas.Login request)
      throws java.rmi.RemoteException;
…
```

In the example, the remote procedure call, login( ), submits a request, passing a Login object as a parameter, and returns a LoginResponse object in response from the BIRT iHub defined by ActuateSoapPort in the SDI.

Using a WSDL binding to generate a Java stub

A Java stub consists of a class containing the proxy code that allows an application to call a remote service as a local object. Using a proxy, a developer does not have to specify the URL, namespace, or parameter arrays that the Service and Call objects require.

The stub converts the call to a Java method to a SOAP call. The stub constructor instantiates the service then adds the references for each qualified name, serializable class, and the JavaBean serialization and deserialization factories to Vectors to complete the implementation of the ActuateSoapPort interface, as shown in the following code:

```
package com.actuate.schemas;

public class ActuateSoapBindingStub extends
   org.apache.axis.client.Stub
   implements com.actuate.schemas.ActuateSoapPort_PortType {
   private java.util.Vector cachedSerClasses = new
   java.util.Vector( );
   private java.util.Vector cachedSerQNames = new
   java.util.Vector( )
   private java.util.Vector cachedSerFactories = new
   java.util.Vector( );
   private java.util.Vector cachedDeserFactories = new
   java.util.Vector( );
…
```

```
public ActuateSoapBindingStub(javax.xml.rpc.Service service)
   throws org.apache.axis.AxisFault {
      if (service == null) {
         super.service = new org.apache.axis.client.Service( );
      } else {
         super.service = service;
      }
…
      qName = new javax.xml.namespace.QName(
         "http://schemas.actuate.com/actuate11", "Login");
         cachedSerQNames.add(qName);
         cls = com.actuate.schemas.Login.class;
         cachedSerClasses.add(cls);
         cachedSerFactories.add(beansf);
         cachedDeserFactories.add(beandf);
```

Implementing the Actuate API service

ActuateAPI interface extends javax.xml.rpc.Service and defines the methods that get the URL for an Actuate SOAP port. The ActuateAPILocator class implements ActuateAPI interface to bind the SOAP port to a physical address. It returns this address using getActuateSoapPortAddress( ), as shown in the following code:

```
package com.actuate.schemas;

public class ActuateAPILocator extends
   org.apache.axis.client.Service
   implements com.actuate.schemas.ActuateAPI {
private final java.lang.String ActuateSoapPort_address = "http://
   ENL2509:8000";

// Use to get a proxy class for ActuateSoapPort
public java.lang.String getActuateSoapPortAddress( ) {
   return ActuateSoapPort_address;
   }
```

ActuateAPI interface specifies two versions of getActuateSoapPort( ) method to access a physical address. ActuateAPILocator.getActuateSoapPort( ) returns the default address set using attribute, ActuateSoapPort_address, as shown in the following code:

```
public com.actuate.schemas.ActuateSoapPort_PortType
   getActuateSoapPort( )
   throws javax.xml.rpc.ServiceException {
   java.net.URL endpoint;
   try {
      endpoint = new java.net.URL(ActuateSoapPort_address);
   }
   catch (java.net.MalformedURLException e) {
      throw new javax.xml.rpc.ServiceException(e);
```

```
   }
   return getActuateSoapPort(endpoint);
}
```

The overloaded version of ActuateAPILocator.getActuateSoapPort(java.net.URL portAddress) accepts a URL as a parameter. This version creates the service using the URL parameter as the endpoint, as shown in the following code:

```
public com.actuate.schemas.ActuateSoapPort_PortType
   getActuateSoapPort
   (java.net.URL portAddress) throws
   javax.xml.rpc.ServiceException {
   try {
      com.actuate.schemas.ActuateSoapBindingStub _stub =
      new com.actuate.schemas.ActuateSoapBindingStub(portAddress,
   this);
      _stub.setPortName(getActuateSoapPortWSDDServiceName( ));
      return _stub;
   }
   catch (org.apache.axis.AxisFault e) {
      return null;
   }
}
```

Developing Actuate Information Delivery API applications

To run the BIRT iHub Integration Technology example applications for the Apache Axis 1.4 client, open a command prompt. Navigate to the Axis Client directory.

In a Windows environment, you can run the file, setClassPath.bat, to set the environment variables needed to access the required source code, compiled classes, libraries, and other resources. setClassPath.bat contains the following environment variable settings:

```
set SAMPLEBASEDIR=.
set LIBDIR=%SAMPLEBASEDIR%\lib
set AXIS_JAR=%LIBDIR%\axis.jar;%LIBDIR%\commons-
   discovery.jar;%LIBDIR%\commons-logging.jar;%LIBDIR%
   \jaxrpc.jar;%LIBDIR%\log4j-1.2.4.jar;%LIBDIR%\wsdl4j.jar
set SUN_JAR=%LIBDIR%\activation.jar;%LIBDIR%\mail.jar;%LIBDIR%
   \saaj.jar
set XMLPARSER_JAR=%LIBDIR%\xercesImpl.jar;%LIBDIR%
   \xmlParserAPIs.jar
set CLASSPATH=%AXIS_JAR%;%SUN_JAR%;%SAMPLEBASEDIR%
   \build;%SAMPLEBASEDIR%\source;%XMLPARSER_JAR%;%LIBDIR%
   \servlet.jar;
```

In a UNIX environment, you can run the shell script, setClassPath.sh. setClassPath.sh contains the following environment variable settings:

```
#!/bin/sh
export SAMPLEBASEDIR
export LIBDIR
export AXIS_JAR
export SUN_JAR
export XMLPARSER_JAR
export CLASSPATH
SAMPLEBASEDIR=`pwd`
LIBDIR=$SAMPLEBASEDIR/lib
AXIS_JAR=$LIBDIR/axis.jar:$LIBDIR/commons-discovery.jar:$LIBDIR
   /commons-logging.jar:$LIBDIR/jaxrpc.jar:$LIBDIR/log4j-
   1.2.4.jar:$LIBDIR/wsdl4j.jar
SUN_JAR=$LIBDIR/activation.jar:$LIBDIR/mail.jar:$LIBDIR/saaj.jar
XMLPARSER_JAR=$LIBDIR/xercesImpl.jar:$LIBDIR/xmlParserAPIs.jar
CLASSPATH=$AXIS_JAR:$SUN_JAR:$SAMPLEBASEDIR/build:$SAMPLEBASEDIR
   /source:$XMLPARSER_JAR:$LIBDIR/servlet.jar
```

The following sections describe the use of the Axis TCPMonitor utility to capture SOAP messages and the development process for following types of Actuate Information Delivery API applications:

- Logging in to an Encyclopedia volume
- Creating a user
- Performing a search operation
- Executing a transaction-based operation
- Uploading a file
- Downloading a file
- Executing a report
- Scheduling a custom event

The code examples and explanations in this chapter parallel the code examples and explanations in the Developing Actuate Information Delivery API applications using Microsoft .NET chapter.

Writing a program that logs in to an Encyclopedia volume

A login operation authenticates a user in an Encyclopedia volume. A login operation involves the following actions:

- An IDAPI application sends a Login request to an Encyclopedia volume.
- The Encyclopedia volume sends a Login response to the IDAPI application.

A Login request receives a Login response message from an Encyclopedia volume. When a Login request succeeds, the Login response message contains an AuthId, which is an encrypted, authenticated token. When a Login request fails, a volume sends a Login response containing an error code and a description of the error.

The authentication ID in the Login response message remains valid for the current session. Any subsequent request that the application sends to an Encyclopedia volume must include the authentication ID in the message.

Each login operation uses the com.actuate.schemas classes to encode and decode the SOAP request and response messages. The following sections describe the SOAP messages, classes, and program interactions necessary to implement a successful login operation.

The following application derives from the code in the BIRT iHub Integration Technology example applications for the Apache Axis 1.4 client. AcLogin class logs in to the Encyclopedia volume to authenticate the user. If the login succeeds, the application prints a success message. If the login fails, the application prints a usage message.

```
import com.actuate.schemas.*;

public class AcLogin {
   public static final String usage =
      "Usage:\n"+
      "     java AcLogin [options]\n";

   public static void main(String[ ] args) {
      // set command line usage
      Arguments.usage = usage;

      // get command line arguments
      Arguments arguments = new Arguments(args);

      try {
         // login to actuate server
         AcController actuateControl = new
            AcController(arguments.getURL( ));
         actuateControl.setUsername(arguments.getUsername( ));
         actuateControl.setPassword(arguments.getPassword( ));
         actuateControl.setTargetVolume(
            arguments.getTargetVolume( ));
         if (actuateControl.login( )) {
            System.out.println("Congratulations!  You have
               successfully logged into Encyclopedia volume as
               "+actuateControl.getUsername( ));
            }
         else {
            System.out.println("Please try again.\n Usage:\n"+"
               java AcLogin [options]\n");
```

```
            }
        }
        catch (Exception e) {
            e.printStackTrace( );
        }

    }
}
```

About the auxiliary classes provided by the sample application

In the code example, the application makes use of the auxiliary classes provided by the BIRT iHub Integration Technology installation for the Apache Axis 1.4 client. These classes perform tasks common to most applications. The source code files for the auxiliary classes are in the source directory. The auxiliary classes are sample application components and are not part of the com.actuate.schemas package generated by the Actuate WSDL document:

- Arguments is an auxiliary class that analyzes the command line arguments to detect predefined options and enumerate any additional arguments. The following list describes the predefined options that can be specified at the command line:
  - serverURL

    -h hostname specifies the SOAP endpoint. The default value, http://localhost:8000, is set in ActuateControl, another auxiliary class.
  - username

    -u username specifies the user name. The default value, Administrator, is set in ActuateControl.
  - password

    -p password specifies the password. The default value, "", is set in ActuateControl.
  - volume

    -v volume specifies the target volume name. Actuate Release 11 requires a volume name in the SOAP header. For earlier releases, this specification is optional.
  - embeddedDownload

    -e turns on the download embedded option. When downloading a file, you can specify whether to embed the content in the response or transmit the file as an attachment. The default value, False, is set in Arguments.
  - printUsage

    -? prints the usage statement.

- ActuateControl is a controller class that handles routine interactions between the client application and the underlying com.actuate.schemas classes. ActuateControl is a sample class. It is not a comprehensive implementation of all possible interactions. It implements the following essential parts of an IDAPI application:
  - Instantiates an ActuateAPILocatorEx object that implements the required Actuate SOAP header extensions and sets the BIRT iHub URL, as shown in the following code example:

    ```
    public ActuateControl( )
       throws MalformedURLException,ServiceException {
       actuateAPI = new ActuateAPILocatorEx( );
       setActuateServerURL(actuateServerURL);
    }
    ```

  - Sets the endpoint for the proxy to the specified value of the Encyclopedia volume URL, as shown in the following code example:

    ```
    public void setActuateServerURL(String serverURL)
       throws MalformedURLException, ServiceException {
       if ((proxy == null) ||
          !serverURL.equals(actuateServerURL)) {
          if (serverURL != null)
          actuateServerURL = serverURL;
          System.out.println("Setting server to " +
             actuateServerURL);
          proxy =
             actuateAPI.getActuateSoapPort(
                new java.net.URL(actuateServerURL));
       }
    }
    ```

  - Creates and configures the Call object that sends the SOAP message to an Encyclopedia volume, as shown in the following code example:

    ```
    public org.apache.axis.client.Call createCall( ) throws
       ServiceException {
       org.apache.axis.client.Call call =
       org.apache.axis.client.Call)
            actuateAPI.createCall( );
       call.setTargetEndpointAddress(this.actuateServerURL);
       return call;
    }
    ```

- ActuateAPIEx interface extends com.actuate.schemas.ActuateAPI. This interface defines the necessary Actuate SOAP header extensions and the Call object that returns the SOAP header element. ActuateAPIEx defines the following Actuate SOAP header extensions:

- AuthId is a system-generated, encrypted String returned by BIRT iHub in a Login response. All requests, except a Login request, must have a valid AuthId in the SOAP header.
- Locale specifies the language, date, time, currency and other conventions for BIRT iHub to use when returning the data to a client.
- TargetVolume indicates the Encyclopedia volume that receives the request.
- ConnectionHandle supports keeping a connection open to view a persistent report. If ConnectionHandle is present in the SOAP header, the system routes subsequent viewing requests to the same View service that returned the ConnectionHandle.
- DelayFlush tells BIRT iHub to write updates to the disk when the value is False.

The following example shows the code for the ActuateAPIEx interface:

```
public interface ActuateAPIEx extends
   com.actuate.schemas.ActuateAPI {
   public void setAuthId(java.lang.String authId);
   public void setLocale(java.lang.String locale);
   public void setTargetVolume(java.lang.String
      targetVolume);
   public void setConnectionHandle(java.lang.String
      connectionHandle);
   public void setDelayFlush(java.lang.Boolean delayFlush);

   public java.lang.String getAuthId( );
   public java.lang.String getLocale( );
   public java.lang.String getTargetVolume( );
   public java.lang.String getConnectionHandle( );
   public java.lang.Boolean getDelayFlush( );

   public org.apache.axis.client.Call getCall( );
}
```

- ActuateAPILocatorEx class extends com.actuate.schemas.ActuateAPILocator and implements the ActuateAPIEx interface. ActuateAPILocatorEx class creates the Call object and adds the Actuate SOAP header, as shown in the following code example:

```
public class ActuateAPILocatorEx
   extends com.actuate.schemas.ActuateAPILocator
   implements ActuateAPIEx {
…
public Call createCall( ) throws ServiceException {
   call = (org.apache.axis.client.Call) super.createCall( );
   if (null != authId)
```

```
      call.addHeader(new SOAPHeaderElement(null, "AuthId",
         authId));
   if (null != locale)
      call.addHeader(new SOAPHeaderElement(null, "Locale",
         locale));
   if (null != targetVolume)
      call.addHeader(
         new SOAPHeaderElement(null, "TargetVolume",
            targetVolume));
   if (null != connectionHandle)
      call.addHeader(
         new SOAPHeaderElement(
            null,
            "ConnectionHandle",
            connectionHandle));
   if (null != targetVolume)
      call.addHeader(
         new SOAPHeaderElement(null, "DelayFlush",
            delayFlush));
   return call;
}
```

Logging in to the Encyclopedia volume

In the example application, AcLogin class calls ActuateControl.login( ) method. This method instantiates the com.actuate.schemas.Login object, then sets the values for username and password, as shown in the following code:

```
public boolean login( ) {
   boolean success = true;
   com.actuate.schemas.Login request =
      new com.actuate.schemas.Login( );
   request.setPassword(password);
   request.setUser(username);
```

ActuateControl.login( ) sets the AuthId to null, then uses the proxy to make a remote procedure call to BIRT iHub. If successful, the call returns a reference to com.actuate.schemas.LoginResponse object, as shown in the following code example:

```
…
   try {
      actuateAPI.setAuthId(null);
      com.actuate.schemas.LoginResponse response =
      proxy.login(request);
      authenticationId = response.getAuthId( );
      actuateAPI.setAuthId(authenticationId);
   }
```

```
   catch (java.rmi.RemoteException e) {
      // login failed
      success = false;
   }
   return success;
}
```

In the code example, the LoginResponse object contains the authentication ID returned by BIRT iHub. The application uses LoginResponse.getAuthId( ) to access the value, then calls ActuateAPIEx.setAuthId( ) to make the value available to the application to embed in the header of a subsequent SOAP request message.

Capturing SOAP messages using Axis TCPMonitor

The Actuate Information Delivery API uses SOAP messaging in a request and response pattern for communications between the client and BIRT iHub System. The Actuate Information Delivery API packages an XML request in a SOAP envelope and sends it to BIRT iHub System through an HTTP connection.

You can use Axis TCPMonitor (tcpmon) utility to monitor the SOAP messages between an application and the Encyclopedia volume. You configure TCPMonitor to listen at a port for an incoming message to the Encyclopedia volume. You redirect the client application to send a request message to the port where TCPMonitor listens.

TCPMonitor intercepts the message, displays the SOAP message in the request panel, then redirects the message to the Encyclopedia volume. When the Encyclopedia volume responds, TCPMonitor intercepts the message, displays the SOAP message in the response panel, then redirects the message to the client application.

TCPMonitor logs each request-response message pair. You can view a message pair by choosing an entry row in the top panel. You can also remove an entry, edit and resend a message, and save a message pair to a file.

The TCPMonitor utility is in the org.apache.axis.utils package in \lib\axis.jar. You can run TCPMonitor from the command line using the following syntax:

```
java org.apache.axis.utils.tcpmon [listenPort targetHost
   targetPort]
```

The following command line statement starts the TCPMonitor graphical user interface:

```
java org.apache.axis.utils.tcpmon
```

In TCPMonitor—Admin, configure the listener port for TCPMonitor, enter the target hostname for BIRT iHub, then enter the target port.

In Figure 6-1, TCPMonitor—Admin sets tcpmon to listen at port 8080, sets the target hostname for BIRT iHub to localhost, and sets the target port to 8000.

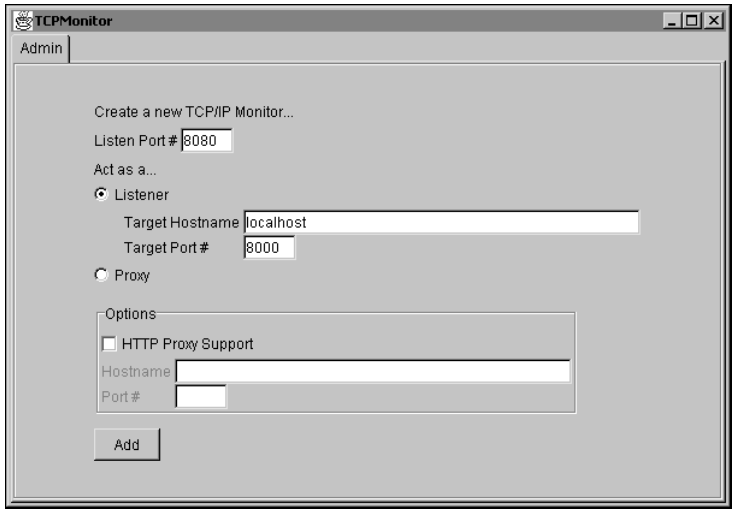


Figure 6-1 The TCP Monitor—Admin page

You can also configure TCPMonitor directly from the command line, as shown in the following example:

```
java org.apache.axis.utils.tcpmon 8080 localhost 8000
```

You can redirect a client application to send a request message to the port where TCPMonitor listens using a command line argument as shown in the following example:

```
java AcLogin -h http://localhost:8080
```

Figure 6-2 shows a request-response message pair from a login operation captured in TCPMonitor.

Writing a simple administration application

A simple administration application typically involves a create operation for one of the following BIRT iHub objects:

- User
- Folder
- Role
- Group

To perform an administration operation that acts on an existing object in the Encyclopedia volume, such as a select, update, or delete operation, you must apply a search condition to the operation. To perform an administration operation that contains a set of administration operations requests, submit the request as a batch or transaction.

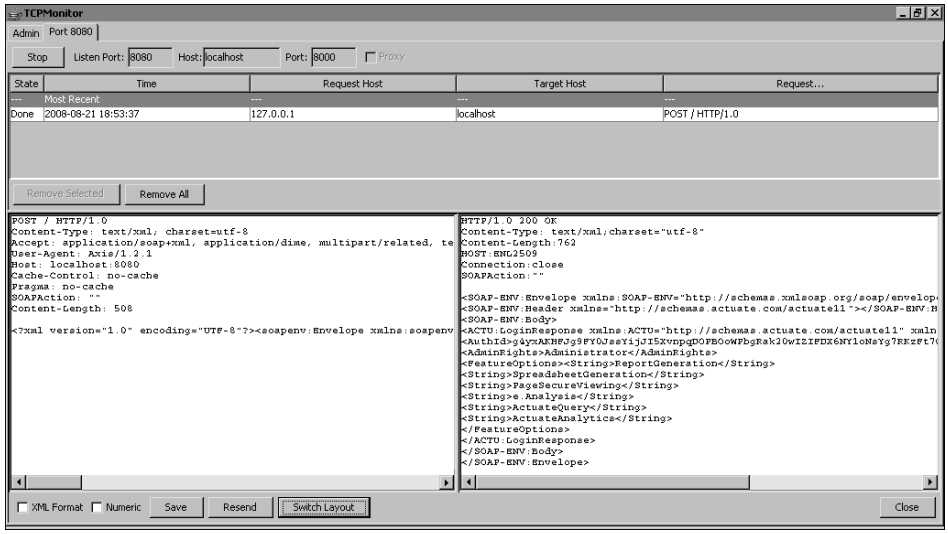


Figure 6-2 Example of a request-response message pair in TCP Monitor

The following sections describe the techniques for building an application that performs a simple administration operation that creates a user in the Encyclopedia volume.

Creating a user

The following application derives from the code in the BIRT iHub Integration Technology example applications for the Apache Axis 1.4 client. The application class, AcCreateUser, logs in to an Encyclopedia volume as Administrator and creates a user. AcCreateUser.createUser( ) method performs the following tasks.

- Instantiates a com.actuate.schemas.User object using ActuateControl.newUser( ) to set the username, password, and home folder:

  ```
  public class AcCreateUser {
  …
     public static void createUser(String userName, String
        homeFolder) throws RemoteException {

           // create a user with password same as userName
           User user = actuateControl.newUser(userName,
           userName, homeFolder);
  ```

  User is a complex data type that represents user attributes.

- Sets additional view preference, notification, e-mail, and job priority options in the User object.

```
// Set user view preference to DHTML
// (defaults to DHTML in standard configuration)
user.setViewPreference(UserViewPreference.DHTML);
   // set notice options
user.setSendNoticeForSuccess(new Boolean(true));
user.setSendNoticeForFailure(new Boolean(true));
// set email options
user.setSendEmailForSuccess(new Boolean(true));
user.setSendEmailForFailure(new Boolean(false));
// create fake email address UserName@localhost
user.setEmailAddress(userName + "@" + "localhost")
   // assign job priority
user.setMaxJobPriority(new Long(1000));
```

- Calls ActuateControl.createUser( ), passing the reference to the User object.

```
   // create the user
   actuateControl.createUser(user);
   System.out.println("User " + userName
      + ", view preferences, send notice and email features,
         plus job priority privileges created.");
   }
}
```

About ActuateControl.createUser( )

ActuateControl.createUser( ) performs the following tasks:

- Instantiates a com.actuate.schemas.CreateUser object.

```
public com.actuate.schemas.AdministrateResponse createUser(
   com.actuate.schemas.User user)
   throws RemoteException {
      com.actuate.schemas.CreateUser createUser =
      new com.actuate.schemas.CreateUser( );
```

  The CreateUser operation is only available to a user with the Administrator role.

- Passes the reference to the User object, containing the username, password, and home folder, and other settings, to the CreateUser object.

```
createUser.setUser(user);
```

- Sets IgnoreDup to True.

```
createUser.setIgnoreDup(Boolean.TRUE);
```

  If the value of IgnoreDup is True, an Encyclopedia volume ignores a duplicate request to create the user and does not report an error. If the value of

IgnoreDup is False, an Encyclopedia volume ignores the duplicate request and reports the error. The default value is False. An Encyclopedia volume always rejects a duplicate request regardless of the IgnoreDup setting.

- Instantiates an AdminOperation object.

```
com.actuate.schemas.AdminOperation adminOperation =
   new com.actuate.schemas.AdminOperation( );
```

An AdminOperation represents a single unit of work within an Administrate request. The list of attributes in the com.actuate.schemas. AdminOperation class lists the possible administration operations that BIRT iHub can perform within the scope of an Actuate Information Delivery API request.

- Passes the reference to the CreateUser object to AdminOperation.setCreateUser( ).

```
adminOperation.setCreateUser(createUser);
```

- Calls ActuateControl.runAdminOperation( ), returning a reference to the AdministrateResponse object that contains the Encyclopedia volume response.

```
      return runAdminOperation(adminOperation);
   }
}
```

About ActuateControl.runAdminOperation( )

ActuateControl.runAdminOperation( ) is an overloaded method that can assemble and run a single administration operation or an array of administration operations. The method that runs a single administration operation performs the following tasks:

- Instantiates an Administrate object.

```
public com.actuate.schemas.AdministrateResponse
   runAdminOperation(
   com.actuate.schemas.AdminOperation adminOperation) {
   com.actuate.schemas.Administrate administrate =
      new com.actuate.schemas.Administrate( );
```

Administrate is not an operation on its own. It is a mechanism for assembling the set of operations that the application is requesting BIRT iHub to perform. An Administrate request can be a composite operation and consist of multiple AdminOperation objects.

- Calls administrate.setAdminOperation( ) to construct the AdminOperation array, adding the AdminOperation object as an element.

```
administrate.setAdminOperation(
   new com.actuate.schemas.AdminOperation[ ] {
   adminOperation });
```

The com.actuate.schemas.Administrate object is an array of AdminOperation objects that can create, update, or delete an item or items in the Encyclopedia volume. An Actuate Information Delivery API application must submit even a single AdminOperation request in an Administrate object as an array of AdminOperations. The AdminOperation array can contain one or more elements.

- Makes the remote call to the Actuate SOAP port using proxy.Administrate( ).

```
com.actuate.schemas.AdministrateResponse administrateResponse =
   null;
try {
   administrateResponse = proxy.administrate(administrate);
```

- Handles a RemoteException.

```
} catch (java.rmi.RemoteException e) {
   org.apache.axis.AxisFault l_fault =
      org.apache.axis.AxisFault.makeFault(e);
   System.out.println(l_fault.getFaultString( ));
   System.out.println(l_fault.getFaultCode().toString( ));
   org.w3c.dom.Element[ ] l_details =
      l_fault.getFaultDetails( );
}
```

- Returns a reference to the com.actuate.schemas.AdministrateResponse object.

```
return administrateResponse;}
```

Performing a search operation

Many operations support acting on one or more items in an Encyclopedia volume. To target the items on which to act, you must apply a search condition to the operation. The com.actuate.schemas library contains many special classes for setting a search condition and implementing a search for an item in an Encyclopedia volume.

The Information Delivery API provides three sets of parameters that support searching for the data on which an operation acts. Typically, these parameters apply to operations that select, update, move, copy, or delete Encyclopedia volume items. The parameters are:

- Id or IdList
- Name or NameList
- Search

Using com.actuate.schemas.SelectFiles

Use com.actuate.schemas.SelectFiles to retrieve file properties using the ID or name of a single file or folder, or a list of files or folders, in an Encyclopedia

volume. SelectFiles can recursively search all subfolders in a directory. To restrict a search to the items in a single directory, not including subfolders, use GetFolderItems. SelectFiles does not retrieve file or folder content.

SelectFiles supports three types of searches:

- Use Name or Id to retrieve a single file or folder.
- Use NameList or IdList to retrieve a list of files or folders in a directory.
- Use Search to retrieve all files or folders that match a given condition.

The following application derives from the code in the BIRT iHub Integration Technology example applications for the Apache Axis 1.4 client. In the example application, AcSelectFiles.searchByCondition( ), calls ActuateControl.selectFiles( ), specifying a search condition, and performing the following operations:

- Specifies a search condition for the file type using the wildcard, *, to target all files that contain the character, R, as the first character in the file extension.

  ```
  public class AcSelectFiles {
     …
     public static String fileType = "R*";
  ```

  A wildcard is a character used in a search or conditional expression that matches one or more literal characters. Actuate wildcards include the ones in the following list:

  - ? matches any single one- or two-byte character.
  - # matches any ASCII numeric character [0-9].
  - * matches any number of characters.

  The wildcard expression in the example targets files in BIRT iHub such as a BIRT executable file with the file extension RPTDESIGN, and a BIRT document file with the file extension RPTDOCUMENT.

- Instantiates a FileCondition object, setting the condition to match on FileField.FileType using the wildcard expression.

  ```
  public static void searchByCondition( ) {
     System.out.println("\nThis example demonstrates search by
        file type using search condition:\n");
     com.actuate.schemas.FileCondition fileCondition =
        new com.actuate.schemas.FileCondition( );
     fileCondition.setField(com.actuate.schemas.FileField.
        FileType);
     fileCondition.setMatch(fileType);
  ```

- Instantiates a FileSearch object, setting the search to the properties specified in the FileCondition object.

```
com.actuate.schemas.FileSearch fileSearch =
   new com.actuate.schemas.FileSearch();
fileSearch.setCondition(fileCondition);
```

An application sets the search condition for a file using the FileSearch class. FileSearch is a complex data type that contains the list of properties to specify in a file search condition. An application can specify the search condition for a file using one or more of the following fields:

- Name
- FileType
- Description
- PageCount
- Size
- TimeStamp
- Version
- VersionName
- Owner

Use ArrayOfFileCondition to specify multiple search conditions.

- Sets up the fetch handle to process the results.

```
fileSearch.setFetchSize(new Integer(2));
System.out.println("Search fileType = " + fileType + "\n");
String fetchHandle = null;
int fetchCount = 1;
while (true) {
   fileSearch.setFetchHandle(fetchHandle);
```

- Makes the call to ActuateControl.selectFiles( ), obtaining the reference to the fetch handle from the SelectFilesResponse object.

```
com.actuate.schemas.SelectFilesResponse selectFilesResponse =
   actuateControl.selectFiles(fileSearch, null, null);
fetchHandle = selectFilesResponse.getFetchHandle( );
```

A fetch handle indicates that the number of items in the result set exceeds the fetch size limit. A fetch handle returns as a parameter in the response to a Select or Get request, such as SelectFiles or GetFolderItems.

Use the fetch handle to retrieve more results from the result set. In the second and subsequent calls for data, you must specify the same search condition that you used in the original call. All Get and Select requests, except SelectFileType, support the use of a fetch handle.

- Continues processing until there are no more results, printing an appropriate output message.

```
      System.out.println("\n  Fetch Size = " +
         fileSearch.getFetchSize( )
          + "; Fetch Count = " + fetchCount++ + "\n");
      if (fetchHandle == null)
         break;
   }
}
```

Using ResultDef

Use a ResultDef parameter to specify the object properties to retrieve from a search. The properties you set in ResultDef depend on the type of item. For example, if the item is a file or folder, you can retrieve the file or folder name, ID, description, date of creation or last update, owner, and version name and number.

ActuateControl.selectFiles( ) specifies and retrieves a list of file properties using a ResultDef parameter by performing the following tasks:

- Sets up the ResultDef String array containing the list of file properties.

```
public com.actuate.schemas.SelectFilesResponse selectFiles(
   com.actuate.schemas.FileSearch fileSearch,
   String name,
   ArrayOfString nameList) {
   com.actuate.schemas.ArrayOfString resultDef =
      newArrayOfString(
         new String[] {
            "Description",
            "FileType",
            "Id",
            "Name",
            "Owner",
            "PageCount",
            "Size",
            "TimeStamp",
            "UserPermissions",
            "Version",
            "VersionName" });
```

- Instantiates the SelectFiles object.

```
com.actuate.schemas.SelectFiles selectFiles =
   new com.actuate.schemas.SelectFiles( );
```

- Selects files based on a file name, list of file names, or uses com.actuate.schemas.ArrayOfString as a ResultDef parameter to specify the file properties to retrieve.

```
selectFiles.setResultDef(resultDef);
```

- If not null, sets the reference to either the search object, file name, or list of file names, depending on which parameter ActuateControl.selectFiles( ) receives.

```
if (fileSearch != null)
   selectFiles.setSearch(fileSearch);
else if (name != null)
   selectFiles.setName(name);
else if (nameList != null)
   selectFiles.setNameList(nameList);
```

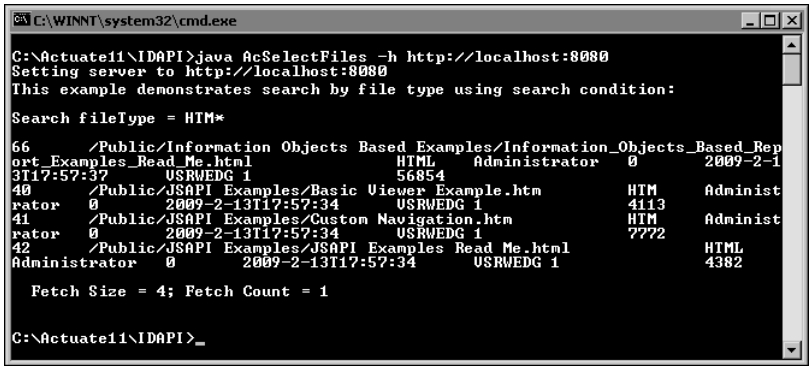
- Makes the com.actuate.schemas.SelectFiles call using the proxy.

```
com.actuate.schemas.SelectFilesResponse selectFilesResponse =
   null;
try {
   selectFilesResponse = proxy.selectFiles(selectFiles);
```

- Loops through the SelectFileResponse item list, printing the file names and list of properties.

```
      com.actuate.schemas.ArrayOfFile itemList =
         selectFilesResponse.getItemList( );
      com.actuate.schemas.File[ ] files = itemList.getFile( );
      if (files != null) {
         for (int i = 0; i < files.length; i++) {
            printFile(System.out, files[i]);
         }
      }
   } catch (RemoteException e) {
      System.out.println("error !!!");
      e.printStackTrace();
   }
   return selectFilesResponse;
}
```

The Java application writes the messages shown in Figure 6-3 to a command prompt window when an Encyclopedia volume login succeeds and the select files operation completes.



```
C:\WINNT\system32\cmd.exe

C:\Actuate11\IDAPI>java AcSelectFiles -h http://localhost:8080
Setting server to http://localhost:8080
This example demonstrates search by file type using search condition:

Search fileType = HTM*

66      /Public/Information Objects Based Examples/Information_Objects_Based_Rep
ort_Examples_Read_Me.html               HTML    Administrator   0       2009-2-1
3T17:57:37      VSRWEDG 1               56854
40      /Public/JSAPI Examples/Basic Viewer Example.htm         HTM     Administ
rator   0       2009-2-13T17:57:34      VSRWEDG 1               4113
41      /Public/JSAPI Examples/Custom Navigation.htm            HTM     Administ
rator   0       2009-2-13T17:57:34      VSRWEDG 1               7772
42      /Public/JSAPI Examples/JSAPI Examples Read Me.html              HTML
Administrator   0       2009-2-13T17:57:34      VSRWEDG 1               4382

  Fetch Size = 4; Fetch Count = 1


C:\Actuate11\IDAPI>_
```

Figure 6-3 Message that shows a successful select files operation

Writing a batch or transaction application

Actuate Information Delivery API supports batch and transaction administration operations. An IDAPI application uses a mixture of developer and com.actuate.schemas classes to implement a batch or transaction administration operation, including:

- Administrate
- AdminOperation
- Transaction
- TransactionOperation

The following sections explain the use of these administration operation classes in detail.

About batch and transaction operations

A batch application submits an array of administration operation requests to an Encyclopedia volume using one composite Administrate message. An Administrate request can contain any number of AdminOperation requests in the batch.

An AdminOperation request can contain any number of Transaction requests. A Transaction request is a composite message that can contain any number of TransactionOperation requests. A TransactionOperation represents a single unit of work within a Transaction.

The default level of granularity for a transaction is an object. One operation run against one object is atomic. The use of an explicit Transaction tag in a composite Administrate message expands the transaction boundary to include multiple TransactionOperation requests.

To perform an administration operation that contains a set of requests, submit the request as a batch or transaction within one composite Administrate message. If a batch request fails, the operations that complete successfully before the failed operation still take effect. If a transaction operation fails, none of the operations in the transaction take effect. BIRT iHub rolls all the work back, leaving the system in the state it was in just prior to the execution of the transaction operation.

Implementing a transaction-based application

The following application derives from the code in the BIRT iHub Integration Technology example applications for the Apache Axis 1.4 client. In the example application, AcAddUsers_Tran.addUsers( ) performs a transaction-based administration operation to add multiple Encyclopedia volume users. When the command line includes the optional ignoreDup argument, the program ignores errors if a user already exists.

AcAddUsers_Tran.addUsers( ) performs the following tasks:

- Defines a TransactionOperation array, dimensioning the array to the number of new users.

```
public class AcAddUsers_Tran {
…
public static void addUsers( ) throws RemoteException {

      // set up transaction operation array
      TransactionOperation[ ] transactionOperations =
         new TransactionOperation[numberOfUsers];

      System.out.println("\nAdding users… \n");
```

- In addUsers( ), within a loop, for each user:
  - Instantiates a com.actuate.schemas.User object, setting the user name, password, home folder, and other options such as view preference, notification, and e-mail.

```
//  begin loop to create a user transaction operation
for (int i = 0; i < numberOfUsers; i++) {

   // set User with user name, password, and home folder
   com.actuate.schemas.User user = new
      com.actuate.schemas.User();
   user.setName(userName);
   user.setPassword(password);
   user.setHomeFolder(homeFolder);
   …
```

  - Instantiates the CreateUser object, passing the reference to the current User object and setting IgnoreDup.

```
// create user
com.actuate.schemas.CreateUser createUser =
   new com.actuate.schemas.CreateUser( );
createUser.setUser(user);
createUser.setIgnoreDup(ignoreDup);
```

- Instantiates an AdminOperation object, passing the reference to the CreateUser object.

```
// set Administration operation for current user
com.actuate.schemas.AdminOperation adminOperation =
   new com.actuate.schemas.AdminOperation( );
adminOperation.setCreateUser(createUser);
```

- Instantiates a TransactionOperation object, passing the reference to the current User object to complete the set up of the transaction operation.

```
// set transaction operation for current user

TransactionOperation transactionOperation =
   new TransactionOperation( );
transactionOperation.setCreateUser(createUser);
```

- Adds the reference to the current TransactionOperation object to the TransactionOperation array.

```
   transactionOperations[i] = transactionOperation;
}
```

- After the end of loop, instantiates a Transaction object, passing the reference to the TransactionOperation array that contains the details of all the create user operations.

```
// set up the transaction with the transaction operations

Transaction transaction = new Transaction( );
transaction.setTransactionOperation(transactionOperations);
```

- Instantiates an AdminOperation object, passing the reference to the Transaction object to create the composite administration operation.

```
// set up Administration operation

AdminOperation adminOperation = new AdminOperation( );
adminOperation.setTransaction(transaction);
```

- Calls ActuateControl.runAdminOperation( ), passing the reference to the composite administration operation, and returning a reference to the AdministrateResponse object that contains the Encyclopedia volume response.

```
// run Administration operation
if (null == actuateControl.runAdminOperation(adminOperation)) {
```

- Prints an output message, indicating the success or failure of the transaction, depending on whether the reference to the AdministrateResponse object is null or valid.

```
        System.out.println("Create user transaction
           failed.\n");
   }
   else {
      System.out.println("Create user transaction succeeded.
         \n");
   }
}
```

Uploading a file

To upload a file to an Encyclopedia volume, you must identify the file to upload and the Encyclopedia volume in which to place the file. You can also specify how to work with existing versions of the file you upload. Using Actuate's open server technology, you can upload third-party file types and native Actuate file types.

About ways of uploading a file

When you upload a file, the content streams to the Encyclopedia volume. You can stream a report with a SOAP message in two ways:

- Embed the file in the response.

  In embedding a file, the application specifies the ContentLength in the HTTP header. If you use HTTP 1.0, you typically choose to embed the file.

- Send the file as a MIME attachment.

  A MIME attachment transmits the contents of the file outside the boundary of the SOAP message.

  A SOAP message with a MIME attachment consists of three parts:

  - HTTP header
  - Actuate SOAP message
  - File attachment

The following example uses a MIME attachment and relevant Actuate Information Delivery API classes to build an application that uploads a file to an Encyclopedia volume.

Using com.actuate.schemas.UploadFile

Use the com.actuate.schemas.UploadFile class to upload a file to an Encyclopedia volume. The file content streams to BIRT iHub as unchunked MIME attachment

to the request. The UploadFile class contains the following attributes:

- NewFile is the com.actuate.schemas.NewFile object to upload.
- CopyFromLatestVersion is an array of Strings used to copy one or more of the following properties from the latest version of the file, if one exists, to the version of the file that you upload:
  - Description is a description of the file.
  - Permissions, the access control list (ACL) specifying the users and roles that can access the file.
  - ArchiveRule specifies the autoarchive rules for the file, which determine how BIRT iHub ages the file and when the file expires.
- Content is the com.actuate.schemas.Attachment object that specifies the content Id, content type, content length, content encoding, locale, and content data.

How to build an application that uploads a file

The following application derives from the code in the BIRT iHub Integration Technology example applications for the Apache Axis 1.4 client. In this example application, AcUploadFile.uploadFile( ) performs the following operations:

- Instantiates a NewFile object, then sets the Encyclopedia volume file location and name

```
public class AcUploadFile {
…
   public static void uploadFile (String locFileName, String
      encycFileName) throws RemoteException,
      ServiceException {
      // set new file information

      com.actuate.schemas.NewFile newFile =
         new com.actuate.schemas.NewFile( );
      newFile.setName(encycFileName);
```

- Instantiates the Attachment object, then sets contentID attribute to the file location and name

```
// set MIME content ID (must be same as AttachmentPart)

com.actuate.schemas.Attachment content =
   new com.actuate.schemas.Attachment( );
content.setContentId(locFileName);
```

- Instantiates the UploadFile object, then passes the references to the NewFile and Attachment objects

```
// set upload message

com.actuate.schemas.UploadFile request =
   new com.actuate.schemas.UploadFile( );
request.setNewFile(newFile);
request.setContent(content);
```

- Sets up the data handler to read the file

```
// use input file as data source

javax.activation.DataHandler dhSource =
   new javax.activation.DataHandler(new
      javax.activation.FileDataSource(locFileName));
```

- Instantiates the org.apache.axis.attachments.AttachmentPart object to contain the data, passes the reference to the data handler, then sets contentID attribute to the file location and name

```
// set attachment in call
org.apache.axis.attachments.AttachmentPart attachmentPart =
   new org.apache.axis.attachments.AttachmentPart( );
attachmentPart.setDataHandler(dhSource);
attachmentPart.setContentId(locFileName);
```

- Performs the UploadFile administration operation by making a call to ActuateControl.uploadFile( )

```
   // call upload file
   com.actuate.schemas.UploadFileResponse response =
      actuateControl.uploadFile(request, attachmentPart);
}
```

ActuateControl.uploadFile( ) performs the following operations:

- Sets up org.apache.axis.client.Call object:
  - Obtains a reference to the Call object using createCall( )

```
public com.actuate.schemas.UploadFileResponse uploadFile(
   com.actuate.schemas.UploadFile request,
   org.apache.axis.attachments.AttachmentPart
      attachmentPart)
   throws RemoteException, RemoteException,
      ServiceException {
   org.apache.axis.client.Call call = createCall( );
```

  - Calls addParameter( ) specifying the following UploadFile parameters:
    - Parameter name
    - XML datatype for the parameter

- Java class for the parameter
- Parameter mode, indicating whether it is ParameterMode.IN, OUT or INOUT

```
call.addParameter(
   new javax.xml.namespace.QName(
      "http://schemas.actuate.com/actuate11",
      "UploadFile"),
   new javax.xml.namespace.QName(
      "http://schemas.actuate.com/actuate11",
      "UploadFile"),
   com.actuate.schemas.UploadFile.class,
   javax.xml.rpc.ParameterMode.IN);
```

- Sets the return type for the operation by specifying parameters that indicate the XML data type and Java class for UploadFileResponse

```
call.setReturnType(
   new javax.xml.namespace.QName(
      "http://schemas.actuate.com/actuate11",
      "UploadFileResponse"));
```

- Sets the UseSOAPAction and SOAPAction URI parameters

```
call.setUseSOAPAction(true);
call.setSOAPActionURI("");
```

- Sets the encoding style URI to null to use the default, binary

```
call.setEncodingStyle(null);
```

- Sets org.apache.axis.AxisEngine.PROP_DOMULTIREFS to False

```
call.setProperty(
   org.apache.axis.AxisEngine.PROP_DOMULTIREFS,
   Boolean.FALSE);
```

PROP_DOMULTIREFS controls the serialization and deserialization processes and the way the client engine handles complex type parameters with multiple references.

- Sets org.apache.axis.client.Call.SEND_TYPE_ATTR to False

```
call.setProperty(
   org.apache.axis.client.Call.SEND_TYPE_ATTR,
   Boolean.FALSE);
```

SEND_TYPE_ATTR controls whether to send xsi type attributes.

- Sets the operation style to document

```
call.setOperationStyle("document");
```

- Converts the operation name String to a QName

```
call.setOperationName(
   new javax.xml.namespace.QName(
      "http://schemas.actuate.com/actuate11",
      "UploadFile"));
```

- Adds the MIME attachment

```
// set the actual MIME attachment
call.addAttachmentPart(attachmentPart);
```

- Makes the call, passing the reference to the UploadFile request in an Object array as required by the Apache Axis framework

```
Object resp = call.invoke(new Object[ ] { request });
```

- If it occurs, handles a RemoteException, otherwise, casts the response into an UploadFileResponse object and returns it to AcUploadFile.uploadFile( )

```
if (resp instanceof java.rmi.RemoteException) {
   throw (java.rmi.RemoteException) resp;
}
   else {
   try {
      return (com.actuate.schemas.UploadFileResponse) resp;
   }
   catch (java.lang.Exception e) {
      return (com.actuate.schemas.UploadFileResponse) org
         .apache
         .axis
         .utils
         .JavaUtils
         .convert(resp,
            com.actuate.schemas.UploadFileResponse.class);
      }
   }
}
```

For more information about the Apache Axis 1.4 client Java-based framework for implementing a SOAP processor, see http://ws.apache.org/axis/java.

Downloading a file

To download a file from an Encyclopedia volume, identify the file and indicate whether to embed the content in the response or use chunked transfer encoding. In HTTP 1.0, you must embed the entire file in the response and send it in a long, uninterrupted file stream. In HTTP 1.1, you can send the file in smaller chunks, which increases the efficiency of the file transfer. Although the Encyclopedia volume supports both methods, BIRT iHub messages typically use chunked transfer encoding.

The following example application uses chunked transfer encoding and relevant com.actuate.schemas classes to build an application that downloads a file from an Encyclopedia volume.

Using com.actuate.schemas.DownloadFile

The com.actuate.schemas.DownloadFile class downloads a file from an Encyclopedia volume to the client. You can choose to embed the file content in the response or send it to the client as an attachment.

The DownloadFile class contains the following list of attributes:

- FileName or FileId is a String specifying the ID or name of the file to download.

  Specify either FileId or FileName.

- DecomposeCompoundDocument is a Boolean indicating whether to download a compound document as one file or multiple attachments.

  If the DecomposeCompoundDocument value is False, you can download the file as a single file. If the value is True, and the file is a compound document, the Encyclopedia volume splits the file into attachments containing the atomic elements of the compound document such as fonts and images. A decomposed document is not in a viewable format. The default value is False.

- DownloadEmbedded is a Boolean indicating whether to embed the content in the response or use chunked transfer encoding.
- FileProperties is a String array specifying the file properties to return.

How to build an application that downloads a file

The following application derives from the code in the BIRT iHub Integration Technology example applications for the Apache Axis 1.4 client. In the example application, AcDownloadFile_Chunked class, downloads a file from the Encyclopedia volume, using the chunked option, and saves the file in the user's ~\temp directory.

The AcDownloadFile_Chunked class performs the following operations:

- Instantiates an org.apache.axis.client.Service object.

```
public class AcDownloadFile_Chunked {
   Service service = new Service( );
```

- In AcDownloadFile_Chunked.main( ), defines an attribute for the file name and initializes the downloadEmbedded flag to False.

```
public static void main(String[ ] args) {
   // download settings
   String  filename;
   Boolean downloadEmbedded = Boolean.FALSE;
```

- Gets the command line argument, specifying the file name.

```
Arguments arguments = new Arguments(args);
filename = arguments.getArgument();
```

- Creates a client Call object, using ActuateControl.createCall( ).

```
try {
   // create a call object
   org.apache.axis.client.Call call =
      actuateControl.createCall( );
```

The ActuateControl.createCall( ) method performs the following tasks:

  - Uses ActuateAPI.createCall( ) to create a client Call object that can send a SOAP message to BIRT iHub
  - Casts the Call object as an org.apache.axis.client.Call object as required by the Apache Axis framework
  - Sets the target BIRT iHub URL
  - Returns the Call object to AcDownloadFile_Chunked.main( )

The following example shows the code for ActuateControl.createCall( ):

```
public org.apache.axis.client.Call createCall( ) throws
   ServiceException {
   org.apache.axis.client.Call call =
      (org.apache.axis.client.Call) actuateAPI.createCall( );
   call.setTargetEndpointAddress(this.actuateServerURL);
   return call;
}
```

- AcDownloadFile_Chunked.main( ) sets up the Call parameters using a local method, prepareDownloadFileCall( ), passing in the reference to the Call object.

```
prepareDownloadFileCall(call);
```

The prepareDownloadFileCall( ) method sets up the org.apache.axis.client.Call parameters as shown in the following code:

```
public static void prepareDownloadFileCall(
   org.apache.axis.client.Call call) {
   call.addParameter(
      new javax.xml.namespace.QName(
         "http://schemas.actuate.com/actuate11",
         "DownloadFile"),
      new javax.xml.namespace.QName(
         "http://schemas.actuate.com/actuate11",
         "DownloadFile"),
```

```
      com.actuate.schemas.DownloadFile.class,
      javax.xml.rpc.ParameterMode.IN);
      call.setReturnType(
      new javax.xml.namespace.QName(
         "http://schemas.actuate.com/actuate11",
         "DownloadFileResponse"));
      call.setUseSOAPAction(true);
      call.setSOAPActionURI("");
      call.setEncodingStyle(null);
      call.setProperty(
         org.apache.axis.AxisEngine.PROP_DOMULTIREFS,
         Boolean.FALSE);
      call.setProperty(snew javax.xml.namespace.QName(
         "http://schemas.actuate.com/actuate11",
         "DownloadFile"));
   }
```

- Next, AcDownloadFile_Chunked.main( ) sets up the DownloadFile request object, specifying the downloadEmbedded flag and file name.

```
// set download message
com.actuate.schemas.DownloadFile request =
   new com.actuate.schemas.DownloadFile( );
request.setDownloadEmbedded(downloadEmbedded);
request.setFileName(filename);
```

- Invokes the Call object, passing the reference to the DownloadFile request in an Object array.

```
Object resp = call.invoke(new Object[ ] {request});
```

- Uses org.apache.axis.client.Call.getMessageContext( ) to obtain the reference to the org.apache.axis.MessageContext object.

```
MessageContext messageContext = call.getMessageContext();
```

- Uses MessageContext.getResponseMessage( ) to obtain the response message.

```
Message message = messageContext.getResponseMessage( );
```

- Uses an Iterator to get the attachments and stream the attachment parts to the Axis default location for the downloaded file in user ~/temp directory.

```
      Iterator iterator = message.getAttachments( );
      while (iterator.hasNext( )) {

         AttachmentPart attachmentPart =
            (AttachmentPart) iterator.next( );
```

```
                try {
                   filename =
                      attachmentPart.getDataHandler( ).getName( );
                   System.out.println("Attachment saved at " +
                      filename);
                }
                catch (SOAPException e) {
                   e.printStackTrace( );
                }
             }

          }
          catch (Exception e) {

             e.printStackTrace( );
          }
       }
    }
```

The Java application writes the following messages, shown in Figure 6-4, to the command prompt window when the download succeeds. Notice that the path and file name in the output message provide the file name and location information for the attachment.

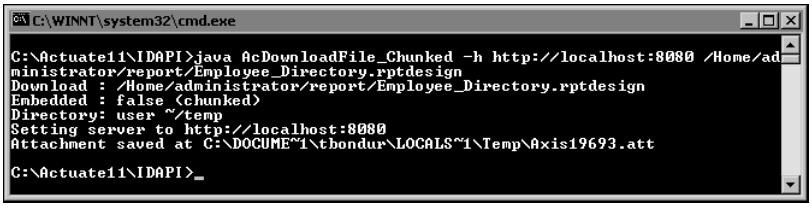


Figure 6-4 Output message for a successful download

Executing a report

An ExecuteReport operation generates a synchronous report from an executable file. The executable file can be an Actuate BIRT file type or an external executable file type.

An ExecuteReport request identifies the input file. To save the output, you indicate the output file's destination by including a full path in the Name parameter for RequestedOutputFile.

An ExecuteReportResponse returns an Encyclopedia volume-generated ObjectId and the status of the report. For a persistent report, the ObjectId is valid until a user deletes the report. For a transient report, the ObjectId is a temporary identifier that lasts for a configurable period of time.

ExecuteReportResponse also returns a ConnectionHandle for a persistent report. Subsequent requests for the same report must use this ConnectionHandle. The ConnectionHandle remains valid throughout the session.

Understanding the structure of an ExecuteReport application

An ExecuteReport application typically uses a mix of developer and com.actuate.schemas classes to implement an ExecuteReport operation in an Encyclopedia volume. The following application derives from the code in the BIRT iHub Integration Technology example applications for the Apache Axis 1.4 client. In the example application, AcExecuteReport, performs these tasks:

- Instantiates the Arguments class, gets the input file (.rptdesign) and output file (.rptdocument) names as command line arguments, and passes these command line arguments to the constructor
- Instantiates the ActuateController class, getting a server URL from Arguments, if specified, at the command line
- Uses methods defined in the controller class to send a request to execute a report using the com.actuate.schemas.ExecuteReport and ExecuteReportResponse classes

The AcExecuteReport class looks like the following example:

```
public class AcExecuteReport {
   public static ActuateControl actuateControl;
   public static void main(String[ ] args) {
      // download settings
      String inputFileName;
      String outputFileName;
      Arguments arguments = new Arguments(args);
      inputFileName = arguments.getArgument( );
      outputFileName = arguments.getArgument( );
      try {
         actuateControl =
            new ActuateControl(arguments.getURL( ));
…
         actuateControl.setInputFileName(inputFileName);
         actuateControl.setOutputFileName(outputFileName);
         actuateControl.executeReport( );
      }
      catch (Exception e) {
         e.printStackTrace( );
      }
   }
}
```

ActuateControl.executeReport( ) performs the following tasks:

- Sets up an ExecuteReport object, specifying the job name, input file name, and output file flag
- Sets up a NewFile object to receive the requested output file
- Calls the proxy to submit the request and receives the response
- Outputs a status message

ActuateControl.executeReport( ) looks like the following example:

```
public void executeReport( ) throws RemoteException {
   com.actuate.schemas.ExecuteReport executeReport =
      new com.actuate.schemas.ExecuteReport( );
   executeReport.setJobName(jobName);
   executeReport.setInputFileName(inputFileName);
   boolean bSaveOutputFile = (!outputFileName.equals(""));
   executeReport.setSaveOutputFile(
      new Boolean(bSaveOutputFile));
   if (bSaveOutputFile) {
      com.actuate.schemas.NewFile requestedOutputFile =
         new com.actuate.schemas.NewFile( );
      requestedOutputFile.setName(outputFileName);
      executeReport.setRequestedOutputFile(requestedOutputFile);
   }
   com.actuate.schemas.ExecuteReportResponse executeReportResponse
      = proxy.executeReport(executeReport);
   System.out.println("Status " +
   executeReportResponse.getStatus( ));
}
```

Using SelectJavaReportPage

SelectJavaReportPage retrieves a specific page from an Actuate BIRT report specifying a page number, a page range, a component, or other search criteria. The following sample application, AcSelectJavaReportPage, selects a single page and saves the result in the specified directory by performing the following tasks:

- Instantiates the Arguments class, passing any command line arguments to the constructor
- Instantiates the ActuateController class, getting a server URL from Arguments, if specified in the command line
- Uses methods defined in the controller class to send a request to select a page using com.actuate.schemas.SelectJavaReportPage, SelectJavaReportPageResponse, ViewParameter, ObjectIdentifier, PageIdentifier, NameValuePair, and ArrayOfNameValuePair classes

- Instantiates the SelectJavaReportPage class and specifies the SelectJavaReportPage operation by performing the following tasks:
  - Sets up an ObjectIdentifier object, specifying the name and type of the file to view
  - Sets up a PageIdentifier object, specifying the page number to view and the page view mode
  - Sets up the references to the ObjectIdentifier and PageIdentifier objects in the SelectJavaReportPage object
  - Sets up the view properties used by the BIRT viewer using NameValuePair and ArrayOfNameValuePair classes

    The view properties include the following settings:

    - SVGFlag
      Specifies whether to use Scalable Vector Graphics (SVG), an XML language used to describe two-dimensional graphics, such as vector graphics shapes, images, and text. This flag is used by the chart engine to determine if SVG chart output is supported.
    - ContextPath
      Specifies a context path relative to the root directory of the web server.
    - MasterPage
      Determines whether to use the master page, which defines the layout for the pages of a report.
    - enableMetaData
      Enables the client to retrieve metadata required to communicate with the service.
    - displayGroupIcon
      Enables the display of group icons for viewing related items.
    - displayFilterIcon
      Enables the display of filter icons for viewing items using options a user selects.
    - viewMode
      Specifies whether the report displays in HTML or PDF.
- Makes the remote call to the Actuate SOAP port using proxy.selectJavaReportPage( ), returning a reference to the com.actuate.schemas.SelectJavaReportPageResponse object
- Sets up the download directory using mkdir( )
- Saves the result in the download directory

- To obtain an image on a page in the document, the application performs the following tasks:
  - Gets the page reference
  - Calls Attachment.getContentData( ) to obtain the page content and sets up a ByteArrayInputStream
  - Calls getEmbed( ), passing in the references to ByteArrayInputStream and ActuateControl objects

    getEmbed( ) performs the following tasks:

    - Sets up an InputStreamReader to iteratively read pattern chunks, line by line, to decode the input stream
    - Compiles the generic image file name, specified in a regular expression, into a pattern instance, which allows the application to use a Matcher object to identify an embedded image
    - Uses Matcher.find( ) to scan a chunk to locate a sequence matching the specified pattern
    - Calls returnId( ) to get the image ID
    - Sets up a GetJavaReportEmbededComponent object, specifying the file name, file type, and a name-value pair indicating the image component ID
    - Sets the response connection handle, the session ID of the object
    - Calls ActuateControl.GetJavaReportEmbededComponent ( ) to get the embedded object
    - Saves the Attachment to the download directory

AcSelectJavaReportPage class looks like the following example:

```
public class AcSelectJavaReportPage {
   public static ActuateControl actuateControl;
   // download settings
   static String filename;
   static String downloadDirectory = "download";
   static String format = "Reportlet";
   static int pageNumber = 1;
   static com.actuate.schemas.SelectJavaReportPageResponse
      resp=null;
   public static void main(String[ ] args)
   {
      // get command line arguments
      Arguments arguments = new Arguments(args);
      filename = arguments.getArgument( );
```

```
   pageNumber =
Integer.parseInt(arguments.getOptionalArgument("1"));
   downloadDirectory =
arguments.getOptionalArgument(downloadDirectory);

   …
      actuateControl.selectPage(
         filename,format,pageNumber,downloadDirectory);
      com.actuate.schemas.SelectJavaReportPage
         acselectjavarptpage = new
            com.actuate.schemas.SelectJavaReportPage( );

      com.actuate.schemas.ObjectIdentifier objId =
         new ObjectIdentifier( );
      objId.setName(filename);
      objId.setType("rptdocument");
      acselectjavarptpage.setObject(objId);

      PageIdentifier pgId = new PageIdentifier( );
      pgId.setPageNum(new Long(pageNumber));
      pgId.setViewMode(new Integer(1));

      acselectjavarptpage.setPage(pgId);
      acselectjavarptpage.setOutputFormat(format);
      acselectjavarptpage.setDownloadEmbedded(
         new Boolean(true));

      com.actuate.schemas.NameValuePair nvpair =
         new com.actuate.schemas.NameValuePair( );
      nvpair0.setName("SVGFlag");
      nvpair0.setValue("false");

      com.actuate.schemas.NameValuePair nvpair1 =
         new com.actuate.schemas.NameValuePair( );
      nvpair1.setName("ContextPath");
      nvpair1.setValue("/");

      com.actuate.schemas.NameValuePair nvpair2 =
         new com.actuate.schemas.NameValuePair( );
      nvpair2.setName("MasterPage");
      nvpair2.setValue("true");

      com.actuate.schemas.NameValuePair nvpair3 =
         new com.actuate.schemas.NameValuePair( );
      nvpair3.setName("enableMetaData");
      nvpair3.setValue("false");

      com.actuate.schemas.NameValuePair nvpair4 =
         new com.actuate.schemas.NameValuePair( );
      nvpair4.setName("displayGroupIcon");
      nvpair4.setValue("false");
```

```
      com.actuate.schemas.NameValuePair nvpair5 =
        new com.actuate.schemas.NameValuePair( );
      nvpair5.setName("displayFilterIcon");
      nvpair5.setValue("false");

      com.actuate.schemas.NameValuePair nvpair6 =
        new com.actuate.schemas.NameValuePair();
      nvpair6.setName("viewMode");
      nvpair6.setValue("1");

      com.actuate.schemas.NameValuePair  npair[ ] =
        {nvpair,nvpair1,nvpair2,nvpair3,
          nvpair4,nvpair5,nvpair6};

      ArrayOfNameValuePair arr = new ArrayOfNameValuePair( );
      arr.setNameValuePair(npair);
      acselectjavarptpage.setViewProperties(arr);

      resp = actuateControl.proxy.selectJavaReportPage(
        acselectjavarptpage);

      // Save the result in download directory
      new java.io.File(downloadDirectory).mkdir( );

      String firstAttachmentName =
        actuateControl.saveAttachment(resp.getPageRef( ),
          downloadDirectory);
      com.actuate.schemas.Attachment att = resp.getPageRef( );

      byte[ ] content = att.getContentData( );
      java.io.ByteArrayInputStream ba =
        new java.io.ByteArrayInputStream(content);

      getembed(ba,actuateControl);
    }
    catch (Exception e)
    {
      e.printStackTrace( );
    }

  public static void getembed(
    java.io.ByteArrayInputStream ba,
      ActuateControl actuateControl) throws Exception {
     Charset cs = Charset.forName("UTF-8");
     java.io.InputStreamReader isr =
        new java.io.InputStreamReader(ba,cs);

     Pattern p = Pattern.compile("__imageID=(\\S+\\.jpg)");

     java.io.BufferedReader br =
        new java.io.BufferedReader(isr);
     String chunk = null;
```

```
        while ((chunk = br.readLine( )) != null) {
          Matcher m = p.matcher(chunk);
          String image_id = null;

          if(m.find( ) ) {
            System.out.println("String read :" + chunk);
            String tmp =  m.group();
            image_id = returnId(tmp);
            String decoded_image_id =
              java.net.URLDecoder.decode(image_id, "UTF-8");
            GetJavaReportEmbededComponent jrptComp =
              new GetJavaReportEmbededComponent( );
            jrptComp.setDownloadEmbedded(new Boolean(true));
            ObjectIdentifier jobj = new ObjectIdentifier( );
            jobj.setName(filename);
            jobj.setType("rptdocument");
            jrptComp.setObject(jobj);
            com.actuate.schemas.NameValuePair jpair =
              new com.actuate.schemas.NameValuePair( );
            jpair.setName("compId");
            jpair.setValue(decoded_image_id);
            com.actuate.schemas.NameValuePair  njpair[ ] =
              {jpair};
            ArrayOfNameValuePair jarr =
              new ArrayOfNameValuePair( );
            jarr.setNameValuePair(njpair);
            jrptComp.setComponent(jarr);
            actuateControl.setConnectionHandle(
              resp.getConnectionHandle( ));
            GetJavaReportEmbededComponentResponse jresp =
            actuateControl.proxy.getJavaReportEmbededComponent(
              jrptComp);
            String imageName =
              actuateControl.saveAttachment(
              jresp.getEmbeddedRef( ), downloadDirectory);
            System.out.println( imageName);
            }
          }
      }
   public static String returnId (String str){
        int startPos;
        startPos = str.indexOf("=");
        String id = str.substring(startPos + 1);
        return id;
     }
}
```

Scheduling a custom event

BIRT iHub provides administrators with a flexible set of options for scheduling when a report runs. A report can be scheduled to run at a specific point in time. At times, running a report can be dependent on other data processing tasks, such as a pending update to a data warehouse or some other decision-support system. Unless these processes complete first, the report can be incomplete or contain out-of-date information. BIRT iHub supports event-based scheduling to facilitate these types of processing requirements.

BIRT iHub supports scheduling a report to run at a specific calendar date and time or when one of the following types of events occurs:

- File

  Specify an operating system file or folder as the event criteria. BIRT iHub runs the event-based job when it finds the file or folder in the specified location.

- Job

  Specify a scheduled job in the Encyclopedia volume as the event criteria. When the scheduled job completes, BIRT iHub runs the event-based job.

- Custom

  Specify a web service that BIRT iHub monitors. BIRT iHub communicates with the web service, continuously polling the service with an event name and parameters. BIRT iHub runs the custom-event based job when the web service returns the specified signal.

File and job events do not require custom programming. A custom event requires you to create the web service and configure BIRT iHub to communicate with the service.

The BIRT iHub installation program installs a default custom event web service and configures the Encyclopedia volume to use the service. BIRT iHub Integration Technology provides a customizable web service application as a sample, including the source code, in the Custom Event Web Service folder.

How to configure a custom event web service

iHub Configuration Console provides a system administration interface for setting up the custom event web service. To set up a custom event web service, perform the following tasks:

1. Log in to the iHub Configuration Console, choose Advanced View, then choose System Volumes.
2. On Volumes—Properties, choose Events.

3. On Events, perform the following tasks:
   1. On Polling, specify the following parameters:
      1. Polling interval

         The amount of time between each polling interval. The default value is 5 minutes.
      2. Polling duration

         The amount of time that BIRT iHub continues polling the web service. BIRT iHub polls for the event status until the event occurs or the event expires. The default value is 300 minutes.
      3. Lag time

         The amount of time that an event occurrence is valid to satisfy an event requirement. For example, if BIRT iHub checks the status of an event with the lag time set to 10 minutes, and the event occurs in that 10-minute interval, the event satisfies the job requirement. If the event occurs after the 10-minute interval elapses, the occurrence does not satisfy the job requirement. The default value is 60 minutes.

      These values apply to all event types. A user can modify these default values in each SubmitJob or UpdateJobSchedule request by resetting the values in the accompanying Event object.
   2. Select Enable custom events.
   3. On Custom event web service configuration, specify the following parameters:
      1. In IP address, type the machine name or IP address of the application server running the custom event service.

         The default name on a Windows machine is localhost.
      2. In SOAP port, type the application server SOAP port used by the custom event service.

         The default port is 8900.
      3. In Context string, type the application server context path used by the custom event service.

         If the event service is in $AC_SERVER_HOME/servletcontainer /webapps/myEvent, the context is /myEvent/servlet/AxisServlet. The default context string is /acevent/servlet/AxisServlet.

      If these parameters are not set, BIRT iHub uses the default values to connect to the sample event service. Figure 6-5 shows the default settings for a custom event configuration on a Windows machine.

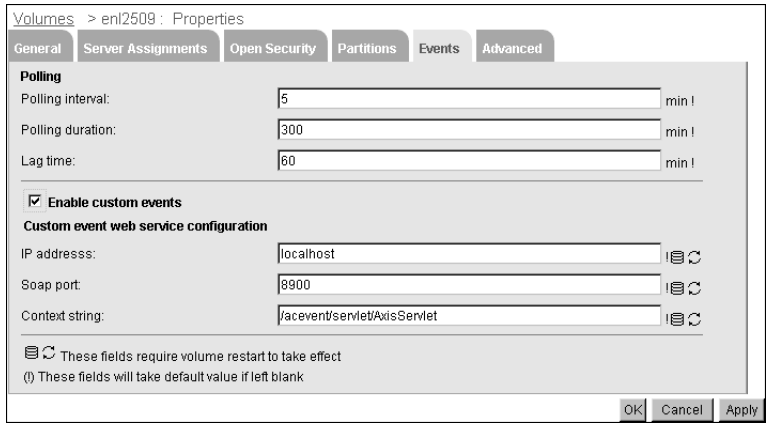


Figure 6-5 Default settings for a custom event configuration on a Windows system

To change the event service configuration, configure these settings in the acserverconfig.xml file:

```
<Volumes>
   <Volume>
         …
         EventPollingInterval="10"
         EventPollingDuration="300"
         EventLagTime=”60”
         CustomEventServiceIPAddress=”hostname or IP address”
         CustomEventServicePort=” 8900 or other port number”
         CustomEventServiceContextString=
            “/myEvent/servlet/AxisServlet”>
         …
   </Volume>
<Volumes>
```

Starting with Release 11 and iHub, the default location for acserverconfig.xml is AC_DATA_HOME/server/config. AC_DATA_HOME refers to the folder the installer specifies as the location for data during the iHub installation. By default, that path is C:/Actuate/iHub/data on a Windows system, and /<Installation directory>/AcServer/data on a Linux system.

How to schedule a custom event

iHub Management Console provides an Encyclopedia volume administration interface for submitting, displaying, and modifying an event-based job.

To specify a custom event when scheduling a job, perform the following tasks:

1. Open iHub Management Console and log in to the Encyclopedia volume as Administrator.

2 On Files and Folders, navigate to the report to run, select the arrow to the left of the report, and choose New Background Job.

3 On Schedule, perform the following tasks, as shown in Figure 6-6:

1 Select Wait for event.

2 In the drop-down box next to Wait for event, select Custom Event.

3 Enter the required event name and parameters.

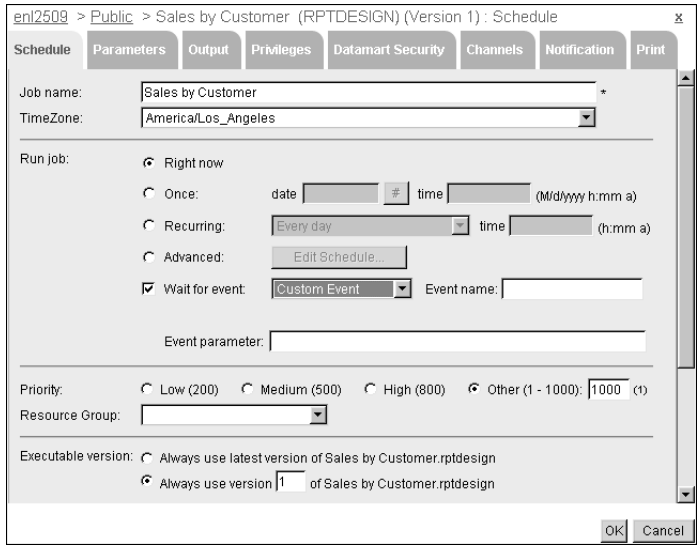


Figure 6-6 Example of job settings for a custom event

Implementing a custom event service

The source code for the sample implementation of a custom event service is in the com.actuate11.event.sample package in the Custom Event Web Service folder of BIRT iHub Integration Technology. This package contains one class, SampleEventService.java. The JAR file for this application is in \lib. The Custom Event Web Service folder also contains a build.xml file for compiling the application using the Apache Ant build tool.

In the installed application, BIRT iHub polls the program using the custom event web service by calling SampleEventService.GetEventStatus( ). This method returns the event status code, which indicates whether the event condition is satisfied or expired.

To implement a custom event service, modify the sample implementation or implement the interface in a class that you create. If you create a class, you must implement the GetEventStatus( ) method of the BIRT iHub EventService interface.

Building a custom event service

BIRT iHub Integration Technology supplies an Apache Ant build script, build.xml. Building the default target "build" using Ant creates eventSample.jar in $INSTALL_DIR\Actuate\ServerIntTech\Custom Event Web Service\lib. Building the target with the "clean" option, cleans up generated class and jar files.

For information about Apache Ant, see the Apache Ant web site at http://ant.apache.org.

Deploying the service on an application server

Stop the application server that runs the event service, add the JAR file to the application server, configure the application server, and restart the server.

For the application server that ships with BIRT iHub, the following directory is the default context for the BIRT iHub custom event service:

```
$AC_SERVER_HOME/servletcontainer/webapps/acevent/WEB-INF/lib
```

To deploy the event service to the default context, copy the JAR file to the lib directory.

Update the event service class.properties file to point to the event service class file. If you deploy the event service using the BIRT iHub default context, update the contents of the class.properties file inside eventWsdl.jar located in the application server directory webapps/acevent/WEB-INF/lib.

The following example shows the setting for the sample event service class:

```
class=com.actuate11.event.sample.SampleEventService
```

As an alternative, you can also create the following application server directory and put the class.properties file in the directory:

```
/webapps/acevent/WEB-INF/classes/com/actuate11/event/wsdl
```

If you use a different context, specify the appropriate context. For example, if you use another folder under webapps called myEvent, then create the following folder myEvent/com/actuate11/event/wsdl/class.properties and point it to your class:

```
class=com.myCompany.myEvent
```

About the custom event web service sample

The SampleEventService class implements the BIRT iHub EventService interface. This interface specifies the GetEventStatus( ) method. You must implement this method with custom logic that provides the current status of each event in the input list.

In GetEventStatus( ), the supplied event service logic is minimal. The method performs the following operations, as shown in the next code example:

- Sets up an array to contain the input list received as an ArrayOfEvent object in the request from BIRT iHub
- Sets up an array to contain the output list sent back as an ArrayOfEventStatus object in the response to BIRT iHub
- Iterates through the input event list, instantiating an EventStatus object for each item in the list, and performing the following operations:
  - Sets the event number taken from the input list
  - Sets the status code by testing to see if the event occurred and setting the status to indicate satisfied or expired
  - Adds the EventStatus object to the output list
  - Returns the ArrayOfEventStatus array in the response to BIRT iHub

```
package com.actuate11.event.sample;
import com.actuate11.event.interfaces.*;
import com.actuate11.event.*;
public class SampleEventService implements EventService
{
   boolean logic = true;
   // Implement the custom event service logic here
   public ArrayOfEventStatus GetEventStatus( ArrayOfEvent
   eventList )
      throws SOAPException
   {
      logic = !logic;
      Event[ ] inputList = eventList.getEvent( );
      ArrayOfEventStatus outputList = new ArrayOfEventStatus( );
      EventStatus[ ] eventStatusList = new
         EventStatus[inputList.length];
      if(inputList == null)
         return null;
      for (int i = 0; i < inputList.length; i++) {
         Event inputEvent = inputList[i];
         EventStatus eventStatus = new EventStatus( );
         eventStatus.setEventNumber(inputEvent.
            getEventNumber( ));
         eventStatus.setStatusCode
            (logic?EventStatus_StatusCode.Satisfied:
            EventStatus_StatusCode.Expired);
         eventStatusList[i] = eventStatus;
      }
      outputList.setEventStatus(eventStatusList);
      return outputList;
   }
}
```

SOAP-based event web service operations and data types

This section describes the SOAP-based event web service operations and data types.

ArrayOfEvent

A complex data type representing an array of events.

Schema

```
<xsd:complexType name="ArrayOfEvent">
   <xsd:sequence>
      <xsd:element name="Event" type="typens:Event"
         maxOccurs="unbounded" minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

ArrayOfEventStatus

A complex data type representing an array of event status.

Schema

```
<xsd:complexType name="ArrayOfEventStatus">
   <xsd:sequence>
      <xsd:element name="EventStatus" type="typens:EventStatus"
         maxOccurs="unbounded" minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Event

A complex data type representing an event.

Schema

```
<xsd:complexType name="Event">
   <xsd:sequence>
      <xsd:element name="EventNumber" type="xsd:long" />
      <xsd:element name="VolumeName" type="xsd:string" />
      <xsd:element name="EventName" type="xsd:string" />
      <xsd:element name="EventParameter" type="xsd:string" />
      <xsd:element name="LagTime" type="xsd:long" minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Elements

EventNumber
The event number.

VolumeName
The volume name for the event.

EventName
The name of the event.

EventParameter
A parameter for the event.

LagTime
The event lag time.

EventStatus

A complex data type representing event status.

Schema

```
<xsd:complexType name="EventStatus">
   <xsd:sequence>
      <xsd:element name="EventNumber" type="xsd:long" />
      <xsd:element name="StatusCode">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="Satisfied" />
               <xsd:enumeration value="Polling" />
               <xsd:enumeration value="Expired" />
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
```

Elements

EventNumber
The event number.

StatusCode
The current status of the event. The event can be Satisfied, Polling, or Expired.

GetEventStatus

Returns the event status code, which indicates whether the event condition is satisfied or expired.

Request schema

```
<xsd:complexType name="GetEventStatus">
   <xsd:sequence>
      <xsd:element name="EventList" type="typens:ArrayOfEvent" />
   </xsd:sequence>
</xsd:complexType>
```

Request elements

EventList
The list of events from which to retrieve the event status.

Response schema

```
<xsd:complexType name="GetEventStatusResponse">
   <xsd:sequence>
      <xsd:element name="EventStatusList"
         type="typens:ArrayOfEventStatus" />
   </xsd:sequence>
</xsd:complexType>
```

Response elements

EventStatusList
The list of event status.

Chapter

7

Developing Actuate Information Delivery API applications using Microsoft .NET

This chapter consists of the following topics:

- About the Microsoft .NET client
- About the Actuate Information Delivery API framework
- Developing Actuate Information Delivery API applications

About the Microsoft .NET client

BIRT iHub System contains WSDL documents that define the Actuate web services schema for the Microsoft C# and Visual Basic development environments. The Microsoft .NET client provides a web services framework for building applications that communicate with BIRT iHub System, using SOAP messaging.

The Actuate Information Delivery API framework uses elements of the Microsoft .NET development platform to support the following features:

- Automatic generation of the Actuate Information Delivery API code library from an Actuate WSDL document

  The library contains the classes, including server proxies, that you can use to write an Actuate Information Delivery API application.

- A SOAP processor that provides automatic encoding and decoding of SOAP messages

A Microsoft .NET client solution containing example projects ships with BIRT iHub Integration Technology. The Microsoft .NET client is in the following directory:

```
\Actuate\ServerIntTech\Web Services\Examples\MS .NET Client
```

The Microsoft .NET Client directory contains the following subdirectories:

- Source

  Contains the example projects.

- Report

  Contains the report files used by the example projects.

- Download

  Stores the files or reports downloaded by the example projects. This directory is initially empty.

- Build

  After building an example project, there are two subdirectories in the build directory, debug and release. Each directory contains copies of the executable files for the project.

There are two solution description files in the MS .NET Client root directory:

- Server Proxy.sln

  Open the server proxy solution first, then build it to generate the proxy DLL. The build process puts the Server Proxy.dll and Server Proxy.pdb files in a build directory for other projects to share and reference.

- Examples.sln

  After you build the server proxy, open the examples solution and build it to compile the example projects. Copy the server proxy files, Server Proxy.dll and Server Proxy.pdb, to the /Debug and /Release subfolders of the example projects.

If changes occur in the WSDL interface, download the latest WSDL file from the BIRT iHub and replace the ActuateAPI.wsdl file in the server proxy solution at the following location:

```
\Actuate\ServerIntTech\Web Services\Examples\MS .NET Client
   \source\Server Proxy\Web References\localhost
```

Rebuild the proxy, then rebuild all the examples. To update the WSDL file using the Microsoft .NET development tool, perform the following tasks:

1. In Solution Explorer, expand Web References.
2. Select localhost, right-click, then choose Update Web Reference, as shown in Figure 7-1.

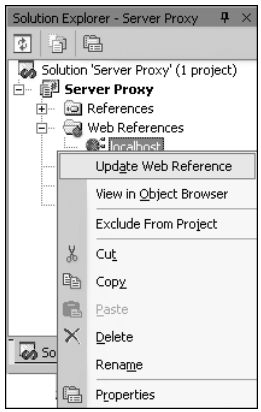


Figure 7-1 Updating the web reference for localhost

This procedure updates the local copy of the WSDL file from an BIRT iHub configured to run at the following default location:

```
http://localhost:8000/wsdl/v11/net/all
```

If you are using a BIRT iHub that listens at a different port on the local machine, change the URL property for the Web Reference, localhost. To update the URL property for the existing Web Reference, localhost, perform the following tasks:

1. In Solution Explorer, expand Web References.
2. Select localhost, right-click, then choose Properties, as shown in Figure 7-2.

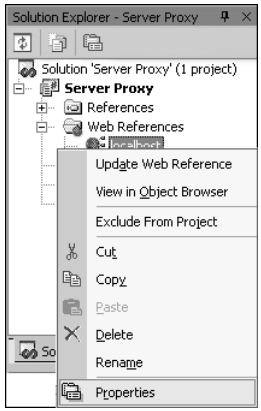


Figure 7-2 Changing the web reference for localhost

Properties appears, as shown in Figure 7-3.

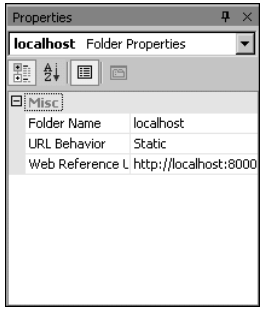


Figure 7-3 The Properties page for localhost

3 Change the Web Reference URL for localhost to the correct value.

Alternatively, you can delete the Web Reference, localhost, then add a new Web Reference that contains the correct URL property.

The \Actuate\ServerIntTech\Web Services\Examples\MS .NET Client \source\Server Proxy\Web References\localhost directory also contains the following files:

- Reference.cs

  Contains the Actuate Information Delivery API classes generated from the Actuate WSDL document

- Reference.map

  Contains the following references:

  - Namespace declaration, xsd, used as a prefix for every Actuate Information Delivery API SOAP message to determine the scope of the XML namespace. The following namespace declaration indicates that a message

is based on the World Wide Web Consortium XML schema initially published in 2001:

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

- Specific instance of the XML schema. In order to use namespace attribute types in an XML document, you must define the xsi namespace, as shown in the following example:

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

- Web services system. The following reference specifies the ActuateAPI.wsdl document for Actuate 11:

```
url="http://localhost:8000/wsdl/v11/net/all"
   filename="ActuateAPI.wsdl"
```

The following sections describe how to build client applications that use the libraries provided by the Actuate Information Delivery API for development in a Microsoft .NET environment.

About the Actuate Information Delivery API framework

The Actuate Information Delivery API framework contains the client-side bindings that the Actuate IDAPI application requires to implement SOAP processing. The SOAP processor serializes, or transforms a remote procedure call by the application into an XML-based SOAP message that asks BIRT iHub to perform a web service. The application sends the request across the network using the HyperText Transfer Protocol (HTTP) transport layer.

BIRT iHub receives the request and deserializes the SOAP message. BIRT iHub performs an appropriate action and sends a response, in the form of a SOAP message, back to the application.

The SOAP processor embedded in the Actuate Information Delivery API framework automates the serialization and deserialization of SOAP messages, relieving the developer of the necessity to program the application at this level.

The following sections describe these key elements of the framework as background information to provide the developer with an understanding of the way the Actuate Information Delivery API framework operates.

Using a data type from a WSDL document to generate a C# class

When you generate the Actuate Information Delivery API source code, Microsoft .NET builds a C# class from each WSDL type definition.

For example, the Login type definition translates the following WSDL into the C# equivalent:

```
<s:complexType name="Login">
   <s:sequence>
      <s:element name="User" type="s:string" />
         <s:element minOccurs="0" name="Password"
            type="s:string" />
         <s:element minOccurs="0" name="EncryptedPwd"
            type="s:string" />
         <s:element minOccurs="0" name="Credentials"
            type="s:base64Binary" />
         <s:element minOccurs="0" name="Domain"
            type="s:string" />
         <s:element minOccurs="0" name="UserSetting"
            type="s:boolean" />
         <s:element minOccurs="0" name="ValidateRoles"
            type="s0:ArrayOfString" />
   </s:sequence>
</s:complexType>
```

The C# class receives the name that appears in the WSDL type definition. The class defines the attributes and data types for each WSDL element. Applying a System.Xml.Serialization class, such as XmlElementAttribute, specifies how the .NET framework serializes and deserializes a C# attribute.

The following code shows the C# class definition for the Login class:

```
[System.Xml.Serialization.XmlTypeAttribute(
   Namespace="http://schemas.actuate.com/actuate11")]
public class Login {
   public string User;
   public string Password;
   public string EncryptedPwd;
      [System.Xml.Serialization.XmlElementAttribute(DataType=
         "base64Binary")]
      public System.Byte[ ] Credentials;
      public string Domain;
      public bool UserSetting;
      [System.Xml.Serialization.XmlIgnoreAttribute( )]
      public bool UserSettingSpecified;
      [System.Xml.Serialization.XmlArrayItemAttribute(
         String", IsNullable=false)]
      public string[] ValidateRoles;
}
```

Mapping the portType to a web service interface

The .NET framework uses the portType and binding in the WSDL document to create the web service interface. The web service interface specifies the input and output messages of the request-response pairs for an operation and the service name and port.

The following WSDL code shows the specification of the input and output messages, Login and LoginResponse, and the binding of this request-response pair to the login operation:

```
<wsdl:portType name="ActuateSoapPort">
   <wsdl:operation name="login">
      <wsdl:input message="tns:Login" />
      <wsdl:output message="tns:LoginResponse" />
   </wsdl:operation>
   …
```

The following WSDL code shows the specification of the service and port names and the binding of the port to a machine address:

```
<wsdl:binding name="ActuateSoapBinding"
   type="tns:ActuateSoapPort">
   <soap:binding transport="http://schemas.xmlsoap.org/soap
      /http" style="document" />
</wsdl:binding>
<wsdl:service name="ActuateAPI">
   <wsdl:port name="ActuateSoapPort"
      binding="tns:ActuateSoapBinding">
      <soap:address location="http://ENL2509:8000" />
   </wsdl:port>
</wsdl:service>
```

An application uses this information to construct an interface to access the operations available from the service using a remote procedure call (RPC). In the following code example, taken from ActuateAPI class, the client application performs the following tasks:

- Sets up the SOAP message.
- The remote procedure call, login( ), submits a request, passing a Login object as a parameter and returns a LoginResponse object in response from the BIRT iHub defined by ActuateSoapPort in the web service interface.

```
[System.Web.Services.Protocols.SoapHeaderAttribute
   ("HeaderValue")]
```

Developing Actuate Information Delivery API applications

The following sections describe the development process for following types of Actuate Information Delivery API applications:

- Logging in to an Encyclopedia volume
- Creating a user
- Performing a search operation
- Executing a transaction-based operation
- Uploading a file
- Downloading a file

The code examples and explanations in the Developing Actuate Information Delivery API using .NET chapter parallel the code examples and explanations in the Developing Actuate Information Delivery API using Java chapter.

Writing a program that logs in to an Encyclopedia volume

A login operation authenticates a user in an Encyclopedia volume. A login operation involves the following actions:

- An IDAPI application sends a Login request to an Encyclopedia volume.
- The Encyclopedia volume sends a Login response to the IDAPI application.

A Login action receives a LoginResponse message from an Encyclopedia volume. When a Login action succeeds, the LoginResponse message contains an AuthId, which is an encrypted, authenticated token. When a Login action fails, an Encyclopedia volume sends a LoginResponse message containing an error code and a description of the error.

The authentication ID in the LoginResponse message remains valid for the current session. Any subsequent request that the application sends to an Encyclopedia volume must include the authentication ID in the message. The following sections describe the SOAP messages, classes, and program interactions necessary to implement a successful login operation.

The following application derives from the code in the BIRT iHub Integration Technology example applications for the MS .NET Client. AcLogin class logs in to the Encyclopedia volume to authenticate the user. If the login succeeds, the application prints a success message. If the login fails, the application prints a usage message. In AcLogin class, Main function performs the following operations:

- Specifies Actuate namespace and calls Usage function, which analyzes the command line arguments as shown in the following code:

```
using System;
using System.Web.Services.Protocols;
using Server_Proxy.localhost;
   namespace Actuate
{
   class AcLogin
   {
      [STAThread]
      static void Main(string[ ] args)
      {
         string l_url;
         string l_userName;
         string l_password;
         string l_volumename;
         if ( Usage(args, out l_url, out l_userName,
            out l_volumename, out l_password) == 0) return;
```

  The following list describes the options that can be specified as command line arguments:

  - serverURL
    -h specifies the SOAP endpoint. The default value is http://localhost:8000.
  - username
    -u specifies the user name. The default value is Administrator.
  - password
    -p specifies the password. The default value is "".
  - volumename
    -v specifies the target volume name. Actuate Release 11 requires a volume name in the SOAP header. For earlier releases, this specification is optional.
  - printUsage
    -? prints the usage statement. The default value is the following string:

    ```
    Usage: Login -h http://localhost:8000 -u username -p password
       -? help
    ```

- Creates an instance of the server proxy and constructs the SOAP header:

```
ActuateAPI l_proxy = new ActuateAPI( );
l_proxy.HeaderValue = new Header( );
l_proxy.Url = l_url;
```

AcLogin uses the following classes defined in References.cs to perform these operations:

- ActuateAPI class is a subclasses of System.Web.Services.Protocols .SoapHttpClientProtocol that specifies the protocol for .NET to use at run time and defines the proxies used to make web service requests.
- Header class is a subclass of System.Web.Services.Protocols.SoapHeader that defines the following Actuate SOAP header extensions:
  - AuthId is a system-generated, encrypted string returned by BIRT iHub in a Login response. All requests, except a Login request, must have a valid AuthId in the SOAP header.
  - Locale specifies the language, date, time, currency, and other conventions for BIRT iHub to use when returning data to a client.
  - TargetVolume indicates the Encyclopedia volume that receives the request.
  - ConnectionHandle supports keeping a connection open to view a persistent report. If ConnectionHandle is present in the SOAP header, the system routes subsequent viewing requests to the same View service that returned the ConnectionHandle.
  - DelayFlush tells BIRT iHub to write updates to the disk when the value is False.

- Instantiates the Login object and prepares the login parameters

```
Server_Proxy.localhost.Login l_req =
   new Server_Proxy.localhost.Login( );
l_req.Password = l_password;
l_req.User = l_userName;
l_req.Domain = l_volumename;
l_req.UserSetting = true;
l_req.UserSettingSpecified = true;
```

- Sends the Login request, handling any exceptions by writing a usage statement to the console

```
LoginResponse l_res;
try
{
   l_res = l_proxy.login(l_req);
}
catch(SoapException e)
{
   Console.WriteLine("Please try again.\n
   Usage: Login -h http://localhost:8000
   -u username -p password -v volumename -? help \n");
```

```
   PrintExceptionDetails((SoapException) e);
   return;
}
```

- Processes a successful Login response by writing a success message to the console and storing AuthId as a SOAP header value for use in the next request

```
Console.WriteLine("Congratulations!  You have successfully
logged into Encyclopedia volume as "+l_userName);
   // Store the authentication id
   l_proxy.HeaderValue.AuthId = l_res.AuthId;
}
```

AcLogin uses the Actuate Information Delivery API classes in References.cs, generated from the Actuate WSDL document. References.cs provides the following code definitions:

- Declares the proxy namespace, XML serialization, and web services protocols for the system

```
namespace Server_Proxy.localhost {
   using System.Diagnostics;
   using System.Xml.Serialization;
   using System;
   using System.Web.Services.Protocols;
   using System.ComponentModel;
   using System.Web.Services;
```

- Defines the ActuateAPI class, which specifies the bindings for the SOAP HTTP client protocol, extended Actuate SOAP header values, and proxy calls

```
[System.Web.Services.WebServiceBindingAttribute(
   Name="ActuateSoapBinding",
   Namespace="http://schemas.actuate.com/actuate11/wsdl")]
public class ActuateAPI
   : System.Web.Services.Protocols.SoapHttpClientProtocol {
public Header HeaderValue;
public ActuateAPI( ) {
   this.Url = "http://ENL2509:8000";
}
/// <remarks/>
[System.Web.Services.Protocols.SoapHeaderAttribute
   ("HeaderValue")]
[System.Web.Services.Protocols.SoapDocumentMethodAttribute(
      "", Use=System.Web.Services.Description.SoapBindingUse
      .Literal,ParameterStyle=
      System.Web.Services.Protocols.SoapParameterStyle.Bare)]
[return:
   System.Xml.Serialization.XmlElementAttribute(
      "LoginResponse",
      Namespace="http://schemas.actuate.com/actuate11")]
```

```
public LoginResponse
   login([System.Xml.Serialization.XmlElementAttribute
      (Namespace="http://schemas.actuate.com/actuate11")] Login
      Login) {
      object[ ] results = this.Invoke("login", new object[ ] {
         Login});
      return ((LoginResponse)(results[0]));
   }
```

Writing a simple administration application

A simple administration application typically involves a create operation for one of the following BIRT iHub objects:

- User
- Folder
- Role
- Group

To perform an administration operation that acts on an existing object in the Encyclopedia volume, such as a select, update, or delete operation, you must apply a search condition to the operation. To perform an administration operation that contains a set of administration operations requests, submit the request as a batch or transaction.

The following application derives from the code in the BIRT iHub Integration Technology example applications for the MS .NET Client. The application class, AcCreateUser, logs in to an Encyclopedia volume as Administrator and creates a user. AcCreateUser performs the following tasks:

- Instantiates a CreateUser object and prepares a create user operation

  ```
  CreateUser l_createUser = new CreateUser( );
     l_createUser.IgnoreDup = true;
     l_createUser.IgnoreDupSpecified = true;
  ```

- Instantiates a User object, setting the username, password, and home folder

  ```
  l_createUser.User = new User( );
     l_createUser.User.Name = l_UserName;
     l_createUser.User.Password = l_password;
     l_createUser.User.HomeFolder = l_homeFolder;
  ```

  User is a complex data type that represents user attributes.

- Sets additional view preference, notification, e-mail, and job priority options in the User object

  ```
  l_createUser.User.ViewPreference = UserViewPreference.DHTML;
  l_createUser.User.ViewPreferenceSpecified = true;
  ```

```
l_createUser.User.SendNoticeForSuccess = true;
l_createUser.User.SendNoticeForSuccessSpecified = true;
l_createUser.User.SendNoticeForFailure = true;
l_createUser.User.SendNoticeForFailureSpecified = true;
l_createUser.User.SendEmailForSuccess = true;
l_createUser.User.SendEmailForSuccessSpecified = true;
l_createUser.User.SendEmailForFailure = true;
l_createUser.User.SendEmailForFailureSpecified = true;
l_createUser.User.EmailAddress = (l_createUserName + "@" +
   "localhost");
l_createUser.User.MaxJobPriority = 1000;
l_createUser.User.MaxJobPrioritySpecified = true;
```

- Instantiates an AdminOperation object and assigns the reference to the CreateUser object to it

```
AdminOperation l_createUserOpt = new AdminOperation( );
l_createUserOpt.Item = l_createUser;
```

- Assembles the create user request in an AdminOperation array

```
AdminOperation[ ] l_adminRequest = new AdminOperation[1];
l_adminRequest[0] = l_createUserOpt;
```

- Makes an administrate request using the proxy, passing in the reference to the AdminOperation array

```
try {
   l_proxy.administrate(l_adminRequest);
}
catch(SoapException e) {
   PrintExceptionDetails(e);
   return;
}
```

Performing a search operation

Many operations support acting on one or more items in an Encyclopedia volume. To target the items on which to act, you must apply a search condition to the operation. The Actuate Information Delivery API library contains many special classes for setting a search condition and implementing a search for an item in an Encyclopedia volume.

The Actuate Information Delivery API provides three sets of parameters that support searching for the data on which an operation acts. Typically, these parameters apply to operations that select, update, move, copy, or delete Encyclopedia volume items. The parameters are:

- Id or IdList

- Name or NameList
- Search

Use SelectFiles to retrieve file properties using the ID or name of a single file or folder, or a list of files or folders, in an Encyclopedia volume. SelectFiles can recursively search all subfolders in a directory. To restrict a search to the items in a single directory, not including subfolders, use GetFolderItems. SelectFiles does not retrieve file or folder content. SelectFiles supports three types of searches:

- Use Name or Id to retrieve a single file or folder.
- Use NameList or IdList to retrieve a list of files or folders in a directory.
- Use Search to retrieve all files or folders that match a given condition.

The following application derives from the code in the BIRT iHub Integration Technology example applications for the MS .NET Client. In the example application, AcSelectFiles specifies a search condition and performs the following operations:

- Specifies a search condition for the file type using the wildcard, *, to target all files that contain the character, R, as the first character in the file extension

  ```
  namespace Actuate
  {
     class AcSelectFiles
     {
        …
           string fileType = "R*";
  ```

  A wildcard is a character used in a search or conditional expression that matches one or more literal characters. Actuate wildcards include the ones in the following list:

  - ? matches any single one- or two-byte character.
  - # matches any ASCII numeric character [0-9].
  - * matches any number of characters.

  The wildcard expression in the example targets files in BIRT iHub such as a BIRT executable file with the file extension RPTDESIGN, and a BIRT document file with the file extension RPTDOCUMENT.

- Instantiates a FileCondition object, setting the condition to match on FileField.FileType using the wildcard expression

  ```
  static void SearchByCondition (string fileType)
  {
     Server_Proxy.localhost.SelectFiles l_req = new
        SelectFiles( );
     FileCondition l_fileCondition = new FileCondition( );
  ```

```
   l_fileCondition.Field = FileField.FileType;
   l_fileCondition.Match = fileType;
```

- Instantiates a FileSearch object, setting the search to the properties specified in the FileCondition object

  ```
  FileSearch l_fileSearch = new FileSearch( );
  l_fileSearch.Item = l_fileCondition;
  ```

  An application sets the search condition for a file using the FileSearch class. FileSearch is a complex data type that contains the list of properties to specify in a file search condition.

  An application can specify the search condition for a file using one or more of the following fields:

  - Name
  - FileType
  - Description
  - PageCount
  - Size
  - TimeStamp
  - Version
  - VersionName
  - Owner

  Use ArrayOfFileCondition to specify multiple search conditions.

- Specifies the fetch handle and SelectFilesResponse object for processing the results

  ```
  l_fileSearch.FetchSize  = 1;
  l_fileSearch.FetchSizeSpecified = true;
  SelectFilesResponse l_res;
  ```

- Makes the selectFiles( ) proxy call, obtaining the reference to the SelectFilesResponse object

  ```
  do
  {
     try
     {
        l_res = l_proxy.selectFiles(l_req);
     }
     catch(Exception e)
     {
        PrintExceptionDetails(e);
  ```

```
      return;
   }
…
```

A fetch handle indicates that the number of items in the result set exceeds the fetch size limit. A fetch handle returns as a parameter in the response to a Select or Get request, such as SelectFiles or GetFolderItems.

Use the fetch handle to retrieve more results from the result set. In the second and subsequent calls for data, you must specify the same search condition that you used in the original call. All Get and Select requests, except SelectFileType, support the use of a fetch handle.

- Continues processing until there are no more results, printing an appropriate output message

```
   File[ ] l_fileList = (File[ ]) l_res.Item;
      if (l_fileList != null)
   {
         for(int i = 0; i < l_fileList.Length; i++)
         {
            Console.WriteLine();
            Console.WriteLine("Item " + i + " Id:" +
               l_fileList[i].Id);
            Console.WriteLine("Item " + i + " Name:" +
               l_fileList[i].Name);
            Console.WriteLine("Item " + i + " Owner:" +
               l_fileList[i].Owner);
            Console.WriteLine("Item " + i + " Description:" +
               l_fileList[i].Description);
            Console.WriteLine("Item " + i + " File Type:" +
               l_fileList[i].FileType);
            Console.WriteLine("Item " + i + " File Size:" +
               l_fileList[i].Size);
         }
      }
   l_fileSearch.FetchHandle = l_res.FetchHandle;
} while(l_res.FetchHandle != null);
```

Writing a batch or transaction application

Actuate Information Delivery API supports batch and transaction administration operations. An IDAPI application uses a mixture of developer and API classes to implement a batch or transaction administration operation, including:

- Administrate
- AdminOperation

- Transaction
- TransactionOperation

The following sections explain the use of these administration operation classes in detail.

About batch and transaction operations

A batch application submits an array of administration operation requests to an Encyclopedia volume using one composite Administrate message. An Administrate request can contain any number of AdminOperation requests in the batch.

An AdminOperation request can contain any number of Transaction requests.

A Transaction request is a composite message that can contain any number of TransactionOperation requests. A TransactionOperation represents a single unit of work within a Transaction.

The default level of granularity for a transaction is an object. One operation run against one object is atomic. The use of an explicit Transaction tag in a composite Administrate message expands the transaction boundary to include multiple TransactionOperation requests.

To perform an administration operation that contains a set of requests, submit the request as a batch or transaction within one composite Administrate message. If a batch request fails, the operations that complete successfully before the failed operation still take effect. If a transaction operation fails, none of the operations in the transaction take effect. BIRT iHub rolls all the work back, leaving the system in the state it was in just prior to the execution of the transaction operation.

Implementing a transaction-based application

The following application derives from the code in the BIRT iHub Integration Technology example applications for the MS .NET Client. In the example application, AcAddUser_Tran performs a transaction-based administration operation to add multiple Encyclopedia volume users. When the command line includes the optional ignoreDup argument, the program ignores errors if a user already exists.

AcAddUsers_Tran.addUsers( ) performs the following tasks:

- Defines a TransactionOperation array, dimensioning the array to the number of new users

```
namespace Actuate
{
   class AcAddUser_Tran
   {
```

```
…
   TransactionOperation[ ] l_transOperation =
      new TransactionOperation[numberOfUsers];
```

- Within a loop, for each user:
  - Instantiates the CreateUser object and sets IgnoreDup

    ```
    CreateUser l_createUser = new CreateUser( );
    l_createUser.IgnoreDup = true;
    l_createUser.IgnoreDupSpecified = true;
    ```

  - Instantiates a User object, passing the reference to the CreateUser object, and setting the user name, password, home folder, and other options such as view preference, notification, and e-mail

    ```
    for (int i = 0; i < numberOfUsers; i++)
    {
       l_createUser.User = new User( );
       l_createUser.User.Name = l_tranUserName;
       l_createUser.User.Password = l_tranPassword;
       l_createUser.User.HomeFolder = l_tranHomeFolder;
       …
    ```

  - Instantiates an AdminOperation object, passing the reference to the CreateUser object

    ```
    AdminOperation l_createUserOpt = new AdminOperation( );
    l_createUserOpt.Item = l_createUser;
    ```

  - Instantiates a TransactionOperation object, passing the reference to the CreateUser object to complete the set up of the transaction operation

    ```
    l_transOperation[i] = new TransactionOperation( );
    l_transOperation[i].Item = l_createUser;
    ```

- After the end of the loop, instantiates an AdminOperation object, passing the reference to the Transaction object to create the composite administration operation

  ```
  AdminOperation l_adminOperation = new AdminOperation( );
  l_adminOperation.Item = l_transOperation;
  ```

- Instantiates an AdminOperation array, passing the reference to the AdminOperation object to complete the assembly of the administration operation request

  ```
  AdminOperation[ ] l_adminRequest = new AdminOperation[1];
  l_adminRequest[0] = l_adminOperation;
  ```

- Makes the administrate proxy call, passing the reference to the administration operation array, handling any Exception that occurs

```
try
{
   l_proxy.administrate(l_adminRequest);
}
catch(Exception e)
{
   Console.WriteLine("Create user transaction failed.\n");
   PrintExceptionDetails(e);
   return;
}
```

- Prints an output message, if the operation succeeds

```
Console.WriteLine("Create user transaction succeeded.\n");
```

Uploading a file

To upload a file to an Encyclopedia volume, you must identify the file to upload and the Encyclopedia volume in which to place the file. You can also specify how to work with existing versions of the file you upload. Using Actuate's open server technology, you can upload third-party file types and native Actuate file types.

About ways of uploading a file

When you upload a file, the content streams to the Encyclopedia volume. You can stream a report with a SOAP message in two ways:

- Embed the file in the response.
  In embedding a file, the application specifies the ContentLength in the HTTP header. If you use HTTP 1.0, you typically choose to embed the file.

- Send the file as a MIME attachment.
  A MIME attachment transmits the contents of the file outside the boundary of the SOAP message.

  A SOAP message with a MIME attachment consists of three parts:

  - HTTP header
  - Actuate SOAP message
  - File attachment

The following example uses a MIME attachment and relevant Actuate Information Delivery API classes to build an application that uploads a file to an Encyclopedia volume.

Using UploadFile

Use the UploadFile class to upload a file to an Encyclopedia volume. The file content streams to BIRT iHub as an unchunked MIME attachment to the SOAP request. The UploadFile class contains the following attributes:

- NewFile is the NewFile object to upload.
- CopyFromLatestVersion is an array of strings used to copy one or more of the following properties from the latest version of the file, if one exists, to the version of the file that you upload:
  - Description is a description of the file.
  - Permissions, the access control list (ACL) specifying the users and roles that can access the file.
  - ArchiveRule specifies the autoarchive rules for the file, which determine how BIRT iHub ages the file and when the file expires.
- Content is the Attachment object that specifies the content Id, content type, content length, content encoding, locale, and content data.

How to build an application that uploads a file

The following application derives from the code in the BIRT iHub Integration Technology example applications for the MS .NET Client. In this example application, AcUploadFile performs the following operations:

- Instantiates an ActuateAPIEx object for specifying the Actuate IDAPI SOAP header extension elements, such as AuthId

```
namespace Actuate
{
   class AcUploadFile
   {
   …
      ActuateAPI l_proxy;

      ActuateAPIEx l_proxyEx= new ActuateAPIEx( );
      l_proxyEx.Url = l_proxy.Url;
      l_proxyEx.HeaderValue = new Header( );
      l_proxyEx.HeaderValue.AuthId =
         l_proxy.HeaderValue.AuthId;
```

- Prepares the UploadFile request by instantiating UploadFile, NewFile, and Attachment objects, and specifying the file name, content type, and content ID

```
UploadFile l_req = new UploadFile( );
l_req.NewFile = new NewFile( );
l_req.NewFile.Name = l_encFileName;
l_req.Content = new Attachment( );
l_req.Content.ContentType = "binary";
l_req.Content.ContentId = "Attachment";
```

- Opens the file for upload by constructing a FileStream object and passing the reference to the ActuateAPIEx object, handling any exception by outputting a message to the console

```
try
{
   ActuateAPIEx.UploadStream =
      new FileStream(l_localFileName, FileMode.Open);
}
catch(Exception e)
{
   Console.WriteLine("Cannot open the file" + e.Message);
   return;
}
```

- Performs the UploadFile administration operation by making an upload file proxy call and closing the file stream after the operation completes

```
UploadFileResponse l_res = null;
   try
   {
      l_res = l_proxyEx.uploadFile(l_req);
   }
   catch(Exception e)
   {
      PrintExceptionDetails(e);
   }
   Console.WriteLine("Uploaded " + l_localFileName + " with
      file id: "
      + l_res.FileId);
   ActuateAPIEx.UploadStream.Close( );
```

Downloading a file

To download a file from an Encyclopedia volume, identify the file and indicate whether to embed the content in the response or use chunked transfer encoding. In HTTP 1.0, you must embed the entire file in the response and send it in a long, uninterrupted file stream. In HTTP 1.1, you can send the file in smaller chunks, which increases the efficiency of the file transfer. Although the Encyclopedia

volume supports both methods, BIRT iHub messages typically use chunked transfer encoding.

The following example application uses chunked transfer encoding and relevant com.actuate.schemas classes to build an application that downloads a file from an Encyclopedia volume.

Using DownloadFile

The DownloadFile class downloads a file from an Encyclopedia volume to the client. You can choose to embed the file content in the response or send it to the client as an attachment.

The DownloadFile class contains the following list of attributes:

- FileName or FileId is a string specifying the ID or name of the file to download.

  Specify either FileName or FileId.

- DecomposeCompoundDocument is a Boolean indicating whether to download a compound document as one file or multiple attachments.

  If the DecomposeCompoundDocument value is False, you can download the file as a single file. If the value is True, and the file is a compound document, the Encyclopedia volume splits the file into attachments containing the atomic elements of the compound document such as fonts and images. A decomposed document is not in a viewable format. The default value is False.

- DownloadEmbedded is a Boolean indicating whether to embed the content in the response or use chunked transfer encoding.

- FileProperties is a string array specifying the file properties to return.

How to build an application that downloads a file

The following application derives from the code in the BIRT iHub Integration Technology example applications for the MS .NET Client. In the example application, AcDownloadFile_Chunked class, downloads a file from the Encyclopedia volume, using the chunked option, and saves the file in the specified directory.

The AcDownloadFile_Chunked class performs the following operations:

- Instantiates an ActuateAPIEx object for specifying the Actuate IDAPI SOAP header extension elements, such as AuthId

```
namespace Actuate
{
   class AcDownloadFile_Chunked
   {
```

```
…
   ActuateAPI l_proxy;

   ActuateAPIEx l_proxyEx= new ActuateAPIEx( );
   l_proxyEx.Url = l_proxy.Url;
   l_proxyEx.HeaderValue = new Header( );
   l_proxyEx.HeaderValue.AuthId =
      l_proxy.HeaderValue.AuthId;
```

- Prepares the DownloadFile request by instantiating DownloadFile and DownloadFileResponse objects, then specifying the file name, item type, and option to download an embedded file

```
DownloadFile l_req = new DownloadFile( );
l_req.Item = l_filename;
l_req.ItemElementName = ItemChoiceType34.FileName;
l_req.DownloadEmbedded = false;
DownloadFileResponse l_res;
```

- Performs the DownloadFile administration operation by making the download file proxy call and handling any exceptions

```
try
{
   l_res = l_proxyEx.downloadFile(l_req);
}
catch(Exception e)
{
   PrintExceptionDetails(e);
   return;
}
```

- Saves the downloaded file to the specified location and closes the file stream

```
FileStream  l_fileStream = new FileStream(l_directory + "\\"
   + l_filename.Substring(l_filename.LastIndexOf("/")+1),
      FileMode.Create);
((MemoryStream)
   ActuateAPIEx.DownloadStream).WriteTo(l_fileStream);
l_fileStream.Close( );
```

Chapter

8

Actuate Information Delivery API operations

This chapter provides reference documentation for Actuate Information Delivery API operations listed in alphabetical order. Each entry includes a general description of the operation, its schema, and a description of its elements.

About the SOAP header

The SOAP header contains authentication data, locale information, and other required and optional data. The SOAP header element is mandatory for calls to the BIRT iHub. Table 8-1 lists all SOAP header elements that the Actuate Information Delivery API uses.

Table 8-1 SOAP header elements

Element	Description
AuthId	The system-generated, encrypted string that the system returns in the Login response when the client logs in using the Actuate Information Delivery API. Required for all requests except Login requests.
ConnectionHandle	An optional element that supports keeping a connection open to view a persistent report. If ConnectionHandle is present in the SOAP header, the system routes subsequent viewing requests to the same View service that returned the ConnectionHandle. If present, BIRT iHub System ignores the value of TargetVolume.
DelayFlush	A Boolean element that tells BIRT iHub to write updates to the disk when the value is False.
FileType	Specifies which type of file the request contains. Specify FileType in the SOAP header for all execute, submit, and view IDAPI requests.
Locale	An optional element that specifies the locale to use for formatting locale-specific information, such as language, date and time, and other locale-specific conventions before returning the data to the client. If the client does not specify another locale, BIRT iHub System uses the client's default locale.
ReportType	An optional element that specifies which type of report to run.
RequestID	A unique value that identifies the SOAP message.
TargetResourceGroup	An optional element that assigns a synchronous report generation request to a specific resource group at run time.
TargetServer	An optional element that refers to the BIRT iHub in a cluster to which to direct the request. Use this element for requests pertaining to system administration tasks, such as GetFactoryServiceJobs and GetFactoryServiceInfo.

Table 8-1 SOAP header elements

Element	Description
TargetVolume	An element that specifies the Encyclopedia volume to which to direct the request. In Release 10, TargetVolume is an optional element. In Release 11 and iHub, Login and other subsequent messages must specify the Encyclopedia volume name using TargetVolume in the SOAP header.

Administrate

Specifies the AdminOperation element which controls abilities to modify an Encyclopedia volume. Only an Encyclopedia volume administrator or a user in the Administrator role uses Administrate operations.

Request schema

```
<xsd:complexType name="Administrate">
   <xsd:sequence>
      <xsd:element name="AdminOperation"
         type="typens:AdminOperation"
         maxOccurs="unbounded" minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Request elements

AdminOperation
Specifies the type of Administrate operation.

Response schema

```
<xsd:complexType name="AdministrateResponse" />
```

AdminOperation

Controls the ability to create, delete, update, copy, and move items within an Encyclopedia volume. An AdminOperation request represents a single unit of work within an Administrate operation. Only an Encyclopedia volume administrator or a user in the Administrator role uses these operations.

Request schema

```
<xsd:complexType name="AdminOperation">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="Transaction" type="typens:Transaction"
            minOccurs="0" />
         <xsd:element name="CreateUser" type="typens:CreateUser"
            minOccurs="0" />
         <xsd:element name="DeleteUser" type="typens:DeleteUser"
            minOccurs="0" />
         <xsd:element name="UndeleteUser"
```

```
   type="typens:UndeleteUser" minOccurs="0"/>
<xsd:element name="UpdateUser" type="typens:UpdateUser"
   minOccurs="0" />
<xsd:element name="CreateGroup" type="typens:CreateGroup"
   minOccurs="0" />
<xsd:element name="DeleteGroup" type="typens:DeleteGroup"
<xsd:element name="UpdateGroup" type="typens:UpdateGroup"
   minOccurs="0" />
<xsd:element name="CreateChannel"
   type="typens:CreateChannel" minOccurs="0" />
<xsd:element name="DeleteChannel"
   type="typens:DeleteChannel" minOccurs="0" />
<xsd:element name="UpdateChannel"
   type="typens:UpdateChannel" minOccurs="0" />
<xsd:element name="CreateRole" type="typens:CreateRole"
   minOccurs="0" />
<xsd:element name="DeleteRole" type="typens:DeleteRole"
   minOccurs="0" />
<xsd:element name="UpdateRole" type="typens:UpdateRole"
   minOccurs="0" />
<xsd:element name="CreateFileType"
   type="typens:CreateFileType" minOccurs="0" />
 <xsd:element name="DeleteFileType"
   type="typens:DeleteFileType" minOccurs="0" />
<xsd:element name="UpdateFileType"
   type="typens:UpdateFileType" minOccurs="0" />
<xsd:element name="CreateFolder"
   type="typens:CreateFolder" minOccurs="0" />
<xsd:element name="DeleteFile" type="typens:DeleteFile"
   minOccurs="0" />
<xsd:element name="MoveFile" type="typens:MoveFile"
   minOccurs="0" />
<xsd:element name="CopyFile" type="typens:CopyFile"
   minOccurs="0" />
<xsd:element name="UpdateFile" type="typens:UpdateFile"
   minOccurs="0" />
<xsd:element name="DeleteJob" type="typens:DeleteJob"
   minOccurs="0" />
<xsd:element name="DeleteJobNotices"
   type="typens:DeleteJobNotices" minOccurs="0" />
<xsd:element name="DeleteJobSchedule"
   type="typens:DeleteJobSchedule" minOccurs="0"/>
<xsd:element name="UpdateJobSchedule"
   type="typens:UpdateJobSchedule" minOccurs="0" />
<xsd:element name="UpdateVolumeProperties"
   type="typens:UpdateVolumeProperties" minOccurs="0" />
 <xsd:element name="UpdateOpenSecurityCache"
```

```
            type="typens:UpdateOpenSecurityCache" minOccurs="0" />
        </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>
```

Request elements

Transaction
A packaging mechanism for Administrate operations. If a failure occurs anywhere in a transaction, all operations in the transaction fail.

CreateUser
Creates a user in the Encyclopedia volume.

DeleteUser
Deletes one or more users.

UndeleteUser
Undeletes one or more previously deleted users within the unit of work.

UpdateUser
Updates user properties in the Encyclopedia volume.

CreateGroup
Creates a notification group in an Encyclopedia volume.

DeleteGroup
Deletes one or more notification groups.

UpdateGroup
Updates notification group properties in the Encyclopedia volume.

CreateChannel
Creates a channel in an Encyclopedia volume.

DeleteChannel
Deletes channels from the Encyclopedia volume.

UpdateChannel
Updates channel properties in the Encyclopedia volume.

CreateRole
Creates a security role in the Encyclopedia volume.

DeleteRole
Deletes one or more security roles.

UpdateRole
Updates security role properties in the Encyclopedia volume.

CreateFileType
Creates a new file type in BIRT iHub.

DeleteFileType
Deletes file types.

UpdateFileType
Updates file type properties in the Encyclopedia volume.

CreateFolder
Creates a folder in an Encyclopedia volume.

DeleteFile
Deletes files or folders from the Encyclopedia volume.

MoveFile
Moves a file or folder from the working directory to a specified target directory in the Encyclopedia volume.

CopyFile
Copies a file or folder in the working directory to a specified target directory.

UpdateFile
Updates file or folder properties in the Encyclopedia volume.

DeleteJob
Deletes one or more jobs.

DeleteJobNotices
Deletes one or more job notices.

DeleteJobSchedule
Deletes a job schedule.

UpdateJobSchedule
Updates a job schedule.

UpdateVolumeProperties
Updates the properties of a specific Encyclopedia volume.

UpdateOpenSecurityCache
Flushes the Encyclopedia volume's open security cache and retrieves new data from an external security source.

CallOpenSecurityLibrary

Calls the Report Server Security Extension (RSSE) API AcRSSEPassThrough function or PassThrough message, which calls the RSSE for general purposes. The application then interprets the value AcRSSEPassThrough or PassThrough returns, along with the return code. The RSSE library registered with BIRT iHub determines the returned value.

Request schema

```
<xsd:complexType name="CallOpenSecurityLibrary">
   <xsd:sequence>
      <xsd:element name="InputParameter" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

InputParameter
The input parameter string.

Response schema

```
<xsd:complexType name="CallOpenSecurityLibraryResponse">
   <xsd:sequence>
      <xsd:element name="OutputParameter" type="xsd:string"/>
      <xsd:element name="ReturnCode" type="xsd:int"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

OutputParameter
The output parameter string.

ReturnCode
The return code.

CancelJob

Terminates a job.

Request schema

```
<xsd:complexType name="CancelJob">
   <xsd:sequence>
      <xsd:element name="JobId" type="xsd:string" />
   </xsd:sequence>
</xsd:complexType>
```

Request elements

JobId
The id of the job to be cancelled.

Response schema

```
<xsd:complexType name="CancelJobResponse">
   <xsd:sequence>
      <xsd:element name="Status" type="typens:CancelJobStatus" />
      <xsd:element name="ErrorDescription" type="xsd:string"
         minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Status
The status of the cancelled job.

ErrorDescription
An error message regarding the cancelled job.

CancelReport

Stops synchronous report execution. Synchronous report execution can be cancelled only after the connection handle is received. ConnectionHandle is a session ID of the object.

The job submitter can cancel a report. Only the system administrator can cancel or get the information about the report of another user.

Request schema

```
<xsd:complexType name="CancelReport">
   <xsd:sequence>
      <xsd:element name="ObjectId" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ObjectId
The object ID of the report to cancel.

Response schema

```
<xsd:complexType name="CancelReportResponse">
   <xsd:sequence>
      <xsd:element name="Status" type="xsd:string" minOccurs="0"/>
      <xsd:element name="ErrorDescription" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Status
The status of the request. One of the following values:

- Succeeded
  The synchronous report generation was successfully cancelled.
- Failed
  The request failed.
- InActive
  The synchronous report generation is complete and cannot be cancelled.

ErrorDescription
A description of any error that occurred.

CloseInfoObject

Closes an information object.

Request schema

```
<xsd:complexType name="CloseInfoObject">
   <xsd:sequence>
      <xsd:element name="DataFetchHandle" type="xsd:string" />
```

```
   </xsd:sequence>
</xsd:complexType>
```

Request elements

DataFetchHandle
The handle to the information object.

Response schema

```
<xsd:complexType name="CloseInfoObjectResponse" />
```

CopyFile

Copies files or folders to a new location. To copy a single file or folder, specify Name or Id. To copy a list of files or folders, specify NameList or IdList. To copy files or folders that match the specified conditions, specify Search.

Request schema

```
<xsd:complexType name="CopyFile">
   <xsd:sequence>
      <xsd:element name="Target" type="xsd:string"/>
         <xsd:choice minOccurs="0">
            <xsd:element name="WorkingFolderName"
               type="xsd:string"/>
            <xsd:element name="WorkingFolderId" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="Recursive" minOccurs="0"/>
      <xsd:choice>
         <xsd:element name="Search" type="typens:FileSearch"/>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="NameList" type="typens:ArrayOfString"/>
         <xsd:element name="Id" type="xsd:string"/>
         <xsd:element name="Name" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="ReplaceExisting" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="MaxVersions" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="LatestVersionOnly" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Target
The new location for the file or folder. The following rules apply:

- If Target is a file, the operation fails if the source contains a folder.
- If Target is a folder that does not exist, a folder is created.

- If Target is a folder and the source contains a single folder, the contents of the source folder are copied to the target folder and merged with target folder contents.
- If Target is a folder and the source contains a single file or multiple files and folders, the source files and folders are copied to the target folders. All source folders are copied as children of the target folder.

WorkingFolderName
The name of the working folder of the file or folder to copy. Specify either WorkingFolderName or WorkingFolderId.

WorkingFolderId
The ID of the working folder of the file or folder to copy. Specify either WorkingFolderId or WorkingFolderName.

Recursive
Specifies whether to search subfolders. If True, the search includes subfolders. The default value is False.

Search
The search condition that specifies which folders or files to copy.

IdList
The list of file or folder IDs to copy. Specify either IdList or NameList.

NameList
The list of file or folder names to copy. Specify either NameList or IdList.

Id
The ID of the single file or folder to copy. Specify either Id or Name.

Name
The name of the single file or folder to copy. Specify either Name or Id.

ReplaceExisting
If True, the copied file replaces the existing file, if one exists. If the existing file has any dependencies, it is not replaced regardless of the ReplaceExisting setting. If False or if the existing file has any dependencies, a new version of the file is created. The default value is True.

MaxVersions
The maximum number of versions to create. MaxVersions applies only for files and is ignored for folders.

LatestVersionOnly
Specifies whether all versions of the file are copied or only the latest version. Used only when a Search tag is specified. If True, only the latest version of the file that matches the search criteria is copied. If False, all versions of the file are copied. The default value is False.

CreateChannel

Creates a channel. CreateChannel is available only to users with the Administrator role.

Request schema

```
<xsd:complexType name="CreateChannel">
   <xsd:sequence>
      <xsd:element name="Channel" type="typens:Channel"/>
      <xsd:element name="IgnoreDup" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Channel
The properties of the channel to create. A name is required.

IgnoreDup
Specifies whether to report an error when creating the channel if one with the same name already exists. BIRT iHub always rejects a duplicate request regardless of the IgnoreDup setting. If True, BIRT iHub does not report an error. If False, BIRT iHub reports an error. The default value is False.

CreateDatabaseConnection

Creates a connection to an Actuate Caching service (ACS) database. The operation returns an error if the database connection already exists.

Request schema

```
<xsd:complexType name="CreateDatabaseConnection">
   <xsd:sequence>
      <xsd:element name="DatabaseConnection"
         type="typens:DatabaseConnectionDefinition"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

DatabaseConnection
Details about the connection object to create.

Response schema

```
<xsd:complexType name="CreateDatabaseConnectionResponse">
   <xsd:sequence>
      <xsd:element name="FileId" type="xsd:string"/>
      <xsd:element name="Warnings" minOccurs="0"
         type="typens:ArrayOfString"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

FileId
The ID of the connection object.

Warnings
Any problems that occur when iHub attempts to connect to the ACS database.

CreateFileType

Adds a file type. Available only to users with the Administrator role.

Request schema

```
<xsd:complexType name="CreateFileType">
   <xsd:sequence>
      <xsd:element name="FileType" type="typens:FileType"/>
      <xsd:element name="IgnoreDup" type="xsd:boolean"
         minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
```

Request elements

FileType
The properties of the file type to add. The following properties are required:

- Name
- Extension
- IsNative
- IsExecutable
- OutputType
- IsPrintable

IgnoreDup
Specifies whether to report an error when creating the file type if one with the same name already exists. BIRT iHub always rejects a duplicate request regardless of the IgnoreDup setting. If True, BIRT iHub does not report an error. If False, BIRT iHub reports an error. The default value is False.

CreateFolder

Creates a folder in an Encyclopedia volume into which you are currently logged in. To create a folder, you must have permission to add folders to the Encyclopedia volume.

Request schema

```
<xsd:complexType name="CreateFolder">
   <xsd:sequence>
      <xsd:choice minOccurs="0">
         <xsd:element name="WorkingFolderName" type="xsd:string"/>
         <xsd:element name="WorkingFolderId" type="xsd:string"/>
      </xsd:choice>
```

```
      <xsd:element name="FolderName" type="xsd:string">
      </xsd:element>
      <xsd:element name="Description" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="IgnoreDup" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

WorkingFolderName
The name of the working folder for the new folder. Specify either WorkingFolderName or WorkingFolderId.

WorkingFolderId
The ID of the working folder for the new folder. Specify either WorkingFolderId or WorkingFolderName.

FolderName
The name of the new folder, relative to the working folder, if specified. If you do not specify a working folder, you must specify a full path.

Description
The description of the folder.

IgnoreDup
Specifies whether to report an error when creating the folder if one with the same name already exists. BIRT iHub always rejects a duplicate request regardless of the IgnoreDup setting. If True, BIRT iHub does not report an error. If False, BIRT iHub reports an error. The default value is False.

CreateGroup

Creates a user group. CreateGroup is available only to users with the Administrator role.

Request schema

```
<xsd:complexType name="CreateGroup">
   <xsd:sequence>
      <xsd:element name="Group" type="typens:Group"/>
      <xsd:element name="IgnoreDup" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Group
The properties of the group to create. Only a name is required. BIRT iHub ignores the ID if it is specified.

IgnoreDup
Specifies whether to report an error when creating the group if one with the same name already exists. BIRT iHub always rejects a duplicate request regardless of the IgnoreDup setting. If True, BIRT iHub does not report an error. If False, BIRT iHub reports an error. The default value is False.

CreateParameterValuesFile

Creates a report object value (.rov) file. To create the ROV based on specified parameters, specify ParameterList. To create the ROV based on an executable file, specify BasedOnFileName or BasedOnFileId. To create the ROV based on another ROV, specify ParameterFile. If you create the ROV based on either an executable or ROV, all parameters must be defined in the based-on file.

CreateParameterValuesFile ignores parameters not defined in the based-on file.

Request schema

```
<xsd:complexType name="CreateParameterValuesFile">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="BasedOnFileName" type="xsd:string"/>
         <xsd:element name="BasedOnFileId" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="ParameterFile" type="typens:NewFile"/>
      <xsd:element name="ParameterValueList"
         type="typens:ArrayOfParameterValue"/>
      <xsd:element name="FileProperties"
         type="typens:ArrayOfString"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

BasedOnFileName
The name of the executable file on which to base the ROV. Specify either BasedOnFileName or BasedOnFileId.

BasedOnFileId
The ID of the executable file on which to base the ROV. Specify either BasedOnFileId or BasedOnFileName.

ParameterFile
The existing ROV on which to base the ROV.

ParameterValueList
The list of parameters on which to base the ROV.

FileProperties
The file properties to return.

Response schema

```
<xsd:complexType name="CreateParameterValuesFileResponse"
   <xsd:sequence>
      <xsd:element name="ParameterValuesFile" type="typens:File"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ParameterValuesFile
The ROV attributes.

CreateResourceGroup

Creates a resource group and specifies its properties.

Request schema

```
<xsd:complexType name="CreateResourceGroup">
   <xsd:sequence>
      <xsd:element name="ResourceGroup"
         type="typens:ResourceGroup"/>
      <xsd:element name="ResourceGroupSettingsList"
         type="typens:ArrayOfResourceGroupSettings"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ResourceGroup
The resource group details.

ResourceGroupSettingsList
The resource group settings.

Response schema

```
<xsd:complexType name="CreateResourceGroupResponse">
   <xsd:sequence/>
</xsd:complexType>
```

CreateRole

Creates a role for security purposes. Available only to users with the Administrator role.

Request schema

```
<xsd:complexType name="CreateRole">
   <xsd:sequence>
      <xsd:element name="Role" type="typens:Role"/>
      <xsd:element name="IgnoreDup" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Role
The properties of the role to create. A name is required.

IgnoreDup
Specifies whether to report an error when creating the role if one with the same name already exists. BIRT iHub always rejects a duplicate request regardless of the IgnoreDup setting. If True, BIRT iHub does not report an error. If False, BIRT iHub reports an error. The default value is False.

CreateUser

Creates a user. Available only to users with the Administrator role.

Request schema

```
<xsd:complexType name="CreateUser">
   <xsd:sequence>
      <xsd:element name="User" type="typens:User"/>
      <xsd:element name="IgnoreDup" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

User
The properties of the user to create. Only a user name is required.

IgnoreDup
Specifies whether to report an error when creating the user if one with the same name already exists. BIRT iHub always rejects a duplicate request regardless of the IgnoreDup setting. If True, BIRT iHub does not report an error. If False, BIRT iHub reports an error. The default value is False.

DataExtraction

Extracts data from a specified object. DataExtraction does not support extraction from multiple components. If multiple components are specified, DataExtraction only extracts the data of the last component.

Request schema

```
<xsd:complexType name="DataExtraction">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
      <xsd:element name="Component"
         type="typens:ArrayOfNameValuePair" minOccurs="0"/>
      <xsd:element name="Properties"
         type="typens:ArrayOfNameValuePair" minOccurs="0"/>
      <xsd:element name="Columns" type="typens:ArrayOfString"
         minOccurs="0"/>
```

```
      <xsd:element name="FilterList"
         type="typens:ArrayOfDataFilterCondition" minOccurs="0"/>
      <xsd:element name="SortColumnList"
         type="typens:ArrayOfDataSortColumn" minOccurs="0"/>
      <xsd:element name="StartRowNumber" type="xsd:int"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The ID of the object from which to extract the data.

Component
The name, display name, or ID, and the value of the component from which to retrieve the data.

Properties
The properties to retrieve.

Columns
The list of column names.

FilterList
The list of available filters.

SortColumnList
The list of columns on which to sort the query.

StartRowNumber
The row number from which to start data extraction.

Response schema

```
<xsd:complexType name="DataExtractionResponse">
   <xsd:sequence>
      <xsd:element name="ResultSetSchema"
         type="typens:ResultSetSchema" minOccurs="0"/>
      <xsd:element name="DataExtractionRef"
         type="typens:Attachment" minOccurs="0"/>
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ResultSetSchema
The complex data type that describes the result set schema.

DataExtractionRef
The reference to the complex data type that describes the object in the attachment and contains the attachment as binary data.

ConnectionHandle
The ID of the report. Supports viewing a report when the report is already in iHub System. Specified in the SOAP header.

DeleteChannel

Deletes channels. To delete a single channel, specify Name or Id. To delete several channels, specify NameList or IdList. To delete channels that match the specified conditions, specify Search.

Request schema

```
<xsd:complexType name="DeleteChannel">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="Search" type="typens:ChannelSearch"/>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="NameList" type="typens:ArrayOfString"/>
         <xsd:element name="Id" type="xsd:string"/>
         <xsd:element name="Name" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="IgnoreMissing" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Search
The search condition that specifies which channels to delete.

IdList
The list of channel IDs to delete. Specify either IdList or NameList.

NameList
The list of channel names to delete. Specify either NameList or IdList.

Id
The ID of the single channel to delete. Specify either Id or Name.

Name
The name of the single channel to delete. Specify either Name or Id.

IgnoreMissing
Specifies what to do if the specified channel does not exist. If True, BIRT iHub ignores the request. If False, the operation reports an error and stops. The default value is True.

DeleteDatabaseConnection

Deletes an Actuate Caching service (ACS) database connection object.

Request schema

```
<xsd:complexType name="DeleteDatabaseConnection">
   <xsd:sequence>
      <xsd:element name="FileName" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

FileName
The name of the database connection object to delete.

Response schema

```
<xsd:complexType name="DeleteDatabaseConnectionResponse"/>
```

DeleteFile

Deletes files or folders. To delete a single file or folder, specify Name or Id. To delete several files or folders, specify NameList or IdList. To delete files or folders that match the specified conditions, specify Search.

Request schema

```
<xsd:complexType name="DeleteFile">
   <xsd:sequence>
      <xsd:choice minOccurs="0">
         <xsd:element name="WorkingFolderId" type="xsd:string"/>
         <xsd:element name="WorkingFolderName" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="Recursive" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="LatestVersionOnly" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:choice>
         <xsd:element name="Search" type="typens:FileSearch"/>
         <xsd:element name="IdList" type="typens:ArrayOfString">
         <xsd:element name="NameList"
            type="typens:ArrayOfString"/>
         <xsd:element name="Id" type="xsd:string"/>
         <xsd:element name="Name" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="IgnoreMissing" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

WorkingFolderId
The ID of the working folder of the file or folder to delete. Specify either WorkingFolderId or WorkingFolderName.

WorkingFolderName
The name of the working folder of the file or folder to delete. Specify either WorkingFolderName or WorkingFolderId.

Recursive
Specifies whether to delete subfolders. If True, subfolders are deleted. If False, only the specified folder is deleted. The default value is False.

LatestVersionOnly
Specifies whether to delete only the latest version of the file. If True, only the latest version of the file is deleted. The default value is False.

Search
The search condition that specifies which folders or files to delete.

IdList
The list of file or folder IDs to delete. Specify either IdList or NameList.

NameList
The list of file or folder names to delete. Specify either NameList or IdList.

Id
The ID of the single file or folder to delete. Specify either Id or Name.

Name
The name of the single file or folder to delete. Specify either Name or Id.

IgnoreMissing
Specifies what to do if the specified file or folder does not exist. If True, BIRT iHub ignores the request. If False, the operation reports an error and stops. The default value is True.

DeleteFileType

Deletes file types. To delete a single file type, specify Name or Id. To delete several file types, specify NameList or IdList.

Request schema

```
<xsd:complexType name="DeleteFileType">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="NameList" type="typens:ArrayOfString"/>
         <xsd:element name="Name" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="IgnoreMissing" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

NameList
The list of file types to delete.

Name
The name of a single file type to delete.

IgnoreMissing
Specifies what to do if the specified file type does not exist. If True, BIRT iHub ignores the request. If False, the operation reports an error and stops. The default value is True.

DeleteGroup

Deletes notification groups. To delete a single group, specify Name or Id. To delete several groups, specify NameList or IdList. To delete groups that match the specified conditions, specify Search.

Request schema

```
<xsd:complexType name="DeleteGroup">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="Search" type="typens:GroupSearch"/>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="NameList" type="typens:ArrayOfString"/>
         <xsd:element name="Id" type="xsd:string"/>
         <xsd:element name="Name" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="IgnoreMissing" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Search
The search condition that specifies which groups to delete.

IdList
The list of group IDs to delete. Specify either IdList or NameList.

NameList
The list of group names to delete. Specify either NameList or IdList.

Id
The ID of the single group to delete. Specify either Id or Name.

Name
The name of the single group to delete. Specify either Name or Id.

IgnoreMissing
Specifies what to do if the specified group does not exist. If True, BIRT iHub ignores the request. If False, the operation reports an error and stops. The default value is True.

DeleteJob

Deletes scheduled, completed, cancelled, or failed jobs. If an instance of a scheduled report is running when the request is submitted, an exception is thrown.

To delete a single job, specify Id. To delete several jobs, specify IdList. To delete jobs that match the specified conditions, specify Search.

```
Request schema
<xsd:complexType name="DeleteJob">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="Search" type="typens:JobSearch"/>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="Id" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="IgnoreMissing" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="IgnoreActiveJob" type="xsd:boolean"
         default=”false” minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Search
The search conditions. If conditions apply to multiple fields, use ConditionArray.

IdList
The list of job IDs to delete.

Id
The ID of the single job to delete.

IgnoreMissing
Specifies what to do if the specified job does not exist. If True, BIRT iHub ignores the request. If False, the operation reports an error and stops. The default value is True.

IgnoreActiveJob
Flag indicating whether to delete a job if it is active.

DeleteJobNotices

Deletes job notices. A user with the Administrator role can delete all job notices. To delete all job notices, do not specify the user or group.

Request schema

```
<xsd:complexType name="DeleteJobNotices">
   <xsd:sequence>
      <xsd:element name="Search" type="typens:JobNoticeSearch"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Search
The search conditions. If conditions apply to multiple fields, use ConditionArray.

DeleteJobSchedule

Deletes a job schedule. If an instance of a scheduled report is running when the request is submitted, an exception is thrown.

To delete a job schedule, specify Id. To delete several jobs, specify IdList. To delete jobs that match the specified conditions, specify Search.

```
Request schema
<xsd:complexType name="DeleteJobSchedule">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="Search"
            type="typens:JobScheduleSearch"/>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="Id" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="IgnoreMissing" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="IgnoreActiveJob" type="xsd:boolean"
         default=”false” minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Search
The search conditions. If conditions apply to multiple fields, use ConditionArray.

IdList
The list of job schedule IDs to delete.

Id
The ID of the single job schedule to delete.

IgnoreMissing
Specifies what to do if the specified job does not exist. If True, BIRT iHub ignores the request. If False, the operation reports an error and stops. The default value is True.

IgnoreActiveJob
Flag indicating whether to delete a job if it is active.

DeleteResourceGroup

Deletes a resource group. You cannot delete a default resource group. If a scheduled job is assigned to a resource group that you delete, the job remains in a pending state when BIRT iHub runs the job. If job is running on a Factory assigned to a resource group that you delete, the job completes.

Request schema

```
<xsd:complexType name="DeleteResourceGroup">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Name
The name of the resource group to delete.

Response schema

```
<xsd:complexType name="DeleteResourceGroupResponse">
   <xsd:sequence/>
</xsd:complexType>
```

DeleteRole

Deletes roles. To delete a single role, specify Name or Id. To delete several roles, specify NameList or IdList. To delete roles that match the specified conditions, specify Search.

Request schema

```
<xsd:complexType name="DeleteRole">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="Search" type="typens:RoleSearch"/>
```

```
            <xsd:element name="IdList" type="typens:ArrayOfString"/>
            <xsd:element name="NameList" type="typens:ArrayOfString"/>
            <xsd:element name="Id" type="xsd:string"/>
            <xsd:element name="Name" type="xsd:string"/>
         </xsd:choice>
         <xsd:element name="IgnoreMissing" type="xsd:boolean"
              minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
```

Request elements

Search
The search condition that specifies which roles to delete.

IdList
The list of role IDs to delete. Specify either IdList or NameList.

NameList
The list of role names to delete. Specify either NameList or IdList.

Id
The ID of the single role to delete. Specify either Id or Name.

Name
The name of the single role to delete. Specify either Name or Id.

IgnoreMissing
Specifies what to do if the specified role does not exist. If True, BIRT iHub ignores the request. If False, the operation reports an error and stops. The default value is True.

DeleteUser

Deletes users. To delete a single user, specify Name or Id. To delete several users, specify NameList or IdList. To delete users that match the specified conditions, specify Search.

Request schema

```
<xsd:complexType name="DeleteUser">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="Search" type="typens:UserSearch"/>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="NameList" type="typens:ArrayOfString"/>
         <xsd:element name="Id" type="xsd:string"/>
         <xsd:element name="Name" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="IgnoreMissing" type="xsd:boolean"
         minOccurs="0"/>
```

```
      <xsd:element name="PurgeUserInfo" type="xsd:boolean"
        minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Search
The search condition that specifies which users to delete.

IdList
The list of user IDs to delete. Specify either IdList or NameList.

NameList
The list of user names to delete. Specify either NameList or IdList.

Id
The ID of the single user to delete. Specify either Id or the Name.

Name
The name of the single user to delete. Specify either Name or Id.

IgnoreMissing
Specifies what to do if the specified user does not exist. If True, BIRT iHub ignores the request. If False, the operation reports an error and stops. The default value is True.

PurgeUserInfo
Purges user information from the system.

DownloadFile

Downloads a file from an Encyclopedia volume. An attachment is included in the response packet, which refers to file content. The file content is streamed back using SOAP attachment.

Request schema

```
<xsd:complexType name="DownloadFile">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="FileName" type="xsd:string"/>
         <xsd:element name="FileId" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="DecomposeCompoundDocument"
         type="xsd:boolean" minOccurs="0"/>
      <xsd:element name="DownloadEmbedded" type="xsd:boolean"
         default="false" minOccurs="0"/>
      <xsd:element name="FileProperties"
         type="typens:ArrayOfString"
         minOccurs="0"/>
```

```
   </xsd:sequence>
</xsd:complexType>
```

Request elements

FileName
The name of the file to download. Specify either FileName or FileId.

FileId
The ID of the file to download. Specify either FileId or FileName.

DecomposeCompoundDocument
Specifies whether to download a compound document as one file or multiple attachments. If False, you can download the file as a single file. If True, and the file is a compound document, BIRT iHub splits the file into several attachments. The default value is False.

DownloadEmbedded
Specifies whether to send the attachment embedded in the body of the SOAP message or in a separate MIME boundary. If True, the response contains the attachment as embedded data. If False, the attachment is in a separate MIME boundary. The default value is False.

FileProperties
The file properties to return.

Response schema

```
<xsd:complexType name="DownloadFileResponse">
   <xsd:sequence>
      <xsd:element name="File" type="typens:File"/>
         <xsd:choice>
            <xsd:element name="Content" type="typens:Attachment"/>
            <xsd:element name="ContainedFiles"
               type="typens:ArrayOfAttachment"/>
         </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

File
The file properties.

Content
The downloaded file in an embedded or chunked file operation.

ContainedFiles
The downloaded set of files in a decomposed compound document operation.

DownloadTransientFile

Downloads transient files. The request requires a FileId and can also indicate whether to decompose a compound document. File content can be attached or embedded in the response.

Request schema

```
<xsd:complexType name="DownloadTransientFile">
   <xsd:sequence>
      <xsd:element name="FileId" type="xsd:string" />
      <xsd:element name="DecomposeCompoundDocument"
         type="xsd:boolean" minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Request elements

FileId
The file ID of the transient file.

DecomposeCompoundDocument
Flag indicating whether to decompose compound documents into separate attachments.

Response schema

```
<xsd:complexType name="DownloadTransientFileResponse">
   <xsd:sequence>
      <xsd:element name="FileId" type="xsd:string" />
      <xsd:choice>
         <xsd:element name="Content" type="typens:Attachment" />
         <xsd:element name="ContainedFiles"
            type="typens:ArrayOfAttachment" />
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

FileId
The file ID.

Content
An attachment containing the file content.

ContainedFiles
An array of any files contained within the transient file.

ExecuteQuery

Reads a query and generates a DOI.

Request schema

```
<xsd:complexType name="ExecuteQuery">
   xsd:sequence>
      <xsd:element name="JobName" type="xsd:string"/>
      <xsd:choice>
         <xsd:element name="InputFileName" type="xsd:string"/>
         <xsd:element name="InputFileId" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="SaveOutputFile" type="xsd:boolean"
         minOccurs="0"/>
```

```
      <xsd:element name="RequestedOutputFile" type="typens:NewFile"
         minOccurs="0"/>
      <xsd:choice minOccurs="0">
         <xsd:element name="QueryFileName" type="xsd:string"/>
         <xsd:element name="QueryFileId" type="xsd:string"/>
         <xsd:element name="Query" type="typens:Query"/>
      </xsd:choice>
      <xsd:element name="ProgressiveViewing" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="RunLatestVersion" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="IsBundled" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="WaitTime" type="xsd:int" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

JobName
The job name.

InputFileName
The name of the file to use for the query.

InputFileId
The ID of the file to use for the query.

SaveOutputFile
Specifies whether the output file is transient or persistent. If True, the output file is transient. If False, the output file is persistent.

RequestedOutputFile
The output file attributes.

QueryFileName
The name of the output file.

QueryFileId
The ID of the output file.

Query
The query the user selected.

ProgressiveViewing
Specifies whether progressive viewing is enabled.

RunLatestVersion
Used only if the input file is a data object values (.dov) file. Specifies whether the DOV is merged with the latest version of the Actuate Basic information object executable (.dox) file. If True, the DOV is merged. If False, the DOV is not merged.

IsBundled
Specifies whether the report is bundled. If True, the report is bundled. If False, the report is not bundled. The default value is False.

WaitTime
The number of seconds BIRT iHub waits before sending a response to the report generation request. Use WaitTime to provide the ability to cancel a synchronous report generation request. The default value is 150 seconds.

Response schema

```
<xsd:complexType name="ExecuteQueryResponse">
   <xsd:sequence>
      <xsd:element name="Status"
         type="typens:ExecuteReportStatus">
      <xsd:element name="ErrorDescription" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="OutputFileType" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ObjectId" type="xsd:string"/>
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ExecutableFileId" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Status
The status of the request. One of the following values:

- Done
- Failed
- FirstPage

ErrorDescription
The description of the error. Returned if Status is Failed.

OutputFileType
The type of the report.

ObjectId
The object ID of the report.

ConnectionHandle
The ID of the report. Supports viewing a report when the report is already in iHub System. Specified in the SOAP header.

ExecutableFileId
The file ID of the executable returned from the query.

ExecuteReport

Triggers the execution of a report in synchronous mode. If you specify WaitTime, BIRT iHub sends a response within the specified time. Otherwise, BIRT iHub sends a response when the request is complete or, if progressive viewing is enabled, when the first page is complete. Specify FileType in the SOAP header for all execute, submit, and view IDAPI requests.

Request schema

```
<xsd:complexType name="ExecuteReport">
   <xsd:sequence>
      <xsd:element name="JobName" type="xsd:string"/>
         <xsd:choice>
            <xsd:element name="InputFileName" type="xsd:string"/>
            <xsd:element name="InputFileId" type="xsd:string"/>
            <xsd:element name="InputFile" type="typens:Attachment"/>
         </xsd:choice>
         <xsd:element name="OutputFormat" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="SaveOutputFile" type="xsd:boolean"
            minOccurs="0"/>
         <xsd:element name="RequestedOutputFile"
            type="typens:NewFile" minOccurs="0"/>
         <xsd:choice minOccurs="0">
            <xsd:element name="ParameterValues"
               type="typens:ArrayOfParameterValue"/>
            <xsd:element name="ParameterValuesFileName"
               type="xsd:string"/>
            <xsd:element name="ParameterValuesFileId"
               type="xsd:string"/>
         </xsd:choice>
         <xsd:element name="IsBundled" type="xsd:boolean"
            minOccurs="0"/>
         <xsd:element name="OpenServerOptions"
            type="typens:OpenServerOptions" minOccurs="0"/>
         <xsd:element name="ProgressiveViewing" type="xsd:boolean"
            minOccurs="0"/>
         <xsd:element name="WaitTime" type="xsd:int"
            minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

JobName
The name of the job.

InputFileName
The name of the input executable file. Specify either InputFileName or InputFileId.

InputFileId
The ID of the input executable file. Specify either InputFileId or InputFileName.

InputFile
Specifies that the input executable file is an attachment in the response. Valid only if the value of SaveOutputFile is False.

OutputFormat
The display format for the output file.

SaveOutputFile
Specifies whether to use the RequestedOutputFile setting. If False, BIRT iHub ignores RequestedOutputFile. If True and RequestedOutputFile is missing, BIRT iHub reports an error.

RequestedOutputFile
The name to use for the output file. Required for persistent jobs.

ParameterValues
The parameter values with which to overwrite the default parameter values.

ParameterValuesFileName
The name of the report object value (.rov) file to create. If specified, BIRT iHub creates a persistent ROV. Otherwise, BIRT iHub creates a temporary ROV. Specify either ParameterValuesFileName or ParameterValuesFileId.

ParameterValuesFileId
The ID of the ROV to create. If specified, BIRT iHub creates a persistent ROV. Otherwise, BIRT iHub creates a temporary ROV. Specify either ParameterValuesFileId or ParameterValuesFileName.

IsBundled
Specifies whether the report is bundled. If True, the report is bundled. If False, the report is not bundled. The default value is False.

OpenServerOptions
The open server options to use. Valid options are:

- KeepWorkingSpace
- DriverTimeout
- PollingInterval

ProgressiveViewing
Specifies whether to enable progressive viewing. True enables progressive viewing. The default value is True.

WaitTime
The number of seconds BIRT iHub waits before sending a response to the report generation request. Use WaitTime to provide the ability to cancel a synchronous report generation request. The default value is 150 seconds.

Response schema

```
<xsd:complexType name="ExecuteReportResponse">
   <xsd:sequence>
      <xsd:element name="Status"
         type="typens:ExecuteReportStatus">
      <xsd:element name="ErrorDescription" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="OutputFileType" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ObjectId" type="xsd:string"/>
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0"/>
      </xsd:sequence>
</xsd:complexType>
```

Response elements

Status
Indicates the execution status of a synchronous job. One of the following values:

- Done
- Failed
- FirstPage
- Pending

ErrorDescription
The description of the error. Returned if Status is Failed.

OutputFileType
Sets the file type for the output file.

ObjectId
The object ID of the report.

ConnectionHandle
The ID of the report. Supports viewing a report when the report is already in iHub System. Specified in the SOAP header.

ExecuteVolumeCommand

Executes predefined Encyclopedia volume control commands.

Request schema

```
<xsd:complexType name="ExecuteVolumeCommand">
   <xsd:sequence>
      <xsd:element name="VolumeName" type="xsd:string"/>
      <xsd:element name="Command">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="StartPartitionPhaseOut"/>
```

```
                <xsd:enumeration value="StartArchive"/>
                <xsd:enumeration value="StopArchive"/>
             </xsd:restriction>
          </xsd:simpleType>
       </xsd:element>
    </xsd:sequence>
 </xsd:complexType>
```

Request elements

VolumeName
The Encyclopedia volume on which to execute the commands.

Command
One or more of the following commands to execute:

- StartPartitionPhaseOut
  Starts the partition phase out.
- StartArchive
  Starts an archive pass.
- StopArchive
  Stops an archive pass if one is currently running. If an archive pass is not currently running, this command returns a Failed status. This command is asynchronous. This means that the call returns without waiting for the archive pass to stop. To find out the status of the archive pass after sending this command, use GetVolumeProperties.

  Only a user with the Operator or Administrator role can issue this command.

Response schema

```
<xsd:complexType name="ExecuteVolumeCommandResponse">
   <xsd:sequence>
      <xsd:element name="Status">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="Succeeded"/>
               <xsd:enumeration value="Failed"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Status
The status of command execution. One of the following values:

- Succeeded
  The command succeeded.
- Failed
  The command failed.

ExtractParameterDefinitionsFromFile

Retrieves the parameter definitions from the specified file.

Request schema

```
<xsd:complexType name="ExtractParameterDefinitionsFromFile>
   <xsd:sequence>
      <xsd:element name="Content" type="typens:Attachment"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Content
The parameter definitions to retrieve.

Response schema

```
<xsd:complexType
      name="ExtractParameterDefinitionsFromFileResponse">
   <xsd:sequence>
         <xsd:element name="ParameterList"
            type="typens:ArrayOfParameterDefinition"/>
      </xsd:sequence>
</xsd:complexType>
```

Response elements

ParameterList
The requested parameter definitions.

ExportParameterDefinitionsToFile

Exports parameter definitions associated with the specified file to a new file.

Request schema

```
<xsd:complexType name="ExportParameterDefinitionsToFile">
   <xsd:sequence>
      <xsd:element name="ParameterList"
         type="typens:ArrayOfParameterDefinition"/>
      <xsd:element name="DownloadEmbedded" type="xsd:boolean"
         default="false" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ParameterList
The parameters to export.

DownloadEmbedded
Specifies whether to send the attachment embedded in the body of the SOAP message or in a separate MIME boundary. If True, the response contains the attachment as embedded data. If False, the attachment is in a separate MIME boundary. The default value is False.

Response schema

```
<xsd:complexType name="ExportParameterDefinitionsToFileResponse">
   <xsd:sequence>
      <xsd:element name="Content" type="typens:Attachment"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Content
The exported parameter definitions.

FetchInfoObjectData

Retrieves data from an information object.

Request schema

```
<xsd:complexType name="FetchInfoObjectData">
   <xsd:sequence>
      <xsd:element name="DataFetchHandle" type="xsd:string" />
   </xsd:sequence>
</xsd:complexType>
```

Request elements

DataFetchHandle
The handle to the information object.

Response schema

```
<xsd:complexType name="FetchInfoObjectDataResponse">
   <xsd:sequence>
      <xsd:choice minOccurs="0">
         <xsd:element name="DataRef" type="typens:Attachment" />
         <xsd:element name="Data" type="typens:InfoObjectData" />
      </xsd:choice>
      <xsd:element name="DataFetchHandle" type="xsd:string"
         minOccurs="0" />
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Response elements

DataRef
An attachment containing the details of the information object data.

Data
The data from the information object.

DataFetchHandle
The handle to the information object.

ConnectionHandle
The ID of the object. Supports viewing objects that are already in the iHub System. Specified in the SOAP header.

GetBookmarks

Retrieves the bookmarks in a report design.

Request schema

```
<xsd:complexType name="GetBookmarks">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The object from which to retrieve the bookmarks.

Response schema

```
<xsd:complexType name="GetBookmarksResponse">
   <xsd:sequence>
      <xsd:element name="BookmarkList"
         type="typens:ArrayOfBookMark"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

BookmarkList
The array of bookmarks.

GetChannelACL

Retrieves the ACL of the specified channel.

Request schema

```
<xsd:complexType name="GetChannelACL">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="ChannelName" type="xsd:string"/>
         <xsd:element name="ChannelId" type="xsd:string"/>
      </xsd:choice>
      <xsd:choice minOccurs="0">
         <xsd:element name="GrantedUserId" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="GrantedUserName" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="GrantedRoleId" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="GrantedRoleName" type="xsd:string"
            minOccurs="0"/>
      </xsd:choice>
      <xsd:element name="FetchSize" type="xsd:int"
         minOccurs="0"/>
```

```
      <xsd:element name="FetchDirection" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="CountLimit" type="xsd:int"
         minOccurs="0"/>
      <xsd:element name="FetchHandle" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ChannelName
The name of the channel for which to retrieve the ACL. Specify either ChannelName or ChannelId.

ChannelId
The ID of the channel for which to retrieve the ACL. Specify either ChannelId or ChannelName.

GrantedUserId
The user ID. Specify either GrantedUserId or GrantedUserName.

GrantedUserName
The user name. Specify either GrantedUserName or GrantedUserId.

GrantedRoleId
The role ID. Specify either GrantedRoleId or GrantedRoleName.

GrantedRoleName
The role name. Specify either GrantedRoleName or GrantedRoleId.

FetchSize
The maximum number of records to retrieve and return in a result set. The default value is 500.

FetchDirection
If True, records are retrieved forward. If False, records are retrieved backward. The default value is True.

CountLimit
The maximum number of records to count. By default, CountLimit is equal to FetchSize. To count all records, set CountLimit to -1.

FetchHandle
Retrieves more items from the result set. In the second and subsequent calls for data, specify the same search criteria as in the original call.

Response schema

```
<xsd:complexType name="GetChannelACLResponse">
   <xsd:sequence>
      <xsd:element name="ACL" type="typens:ArrayOfPermission"
         minOccurs="0"/>
```

```
      <xsd:element name="FetchHandle" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="TotalCount" type="xsd:int" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ACL
The ACL of the channel.

FetchHandle
Indicates that the number of items in the result set exceeds the FetchSize limit.

TotalCount
The number of entries in the search result set.

GetConnectionPropertyAssignees

Retrieves the users and roles for a file.

Request Schema

```
<xsd:complexType name="GetConnectionPropertyAssignees">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="FileName" type="xsd:string"
            minOccurs="0" />
          <xsd:element name="FileId" type="xsd:string"
            minOccurs="0" />
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

FileName
The name of the file.

FileId
The file ID of the file.

Response schema

```
<xsd:complexType name="GetConnectionPropertyAssigneesResponse">
   <xsd:sequence>
      <xsd:element name="UserNames" type="typens:ArrayOfString" />
      <xsd:element name="RoleNames" type="typens:ArrayOfString" />
   </xsd:sequence>
</xsd:complexType>
```

Response elements

UserNames
The user names attached to the file.

RoleNames
The roles of the associated user names.

GetContent

Retrieves the contents of the specified report component. Specify FileType in the SOAP header for all execute, submit, and view IDAPI requests.

The response to GetContent contains the following data:

- The SOAP response.
- The attachment containing the data.
- If the request input format is Reportlet, an XML response as an attachment is retrieved. The response contains the post process data.

Request schema

```
<xsd:complexType name="GetContent">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
      <xsd:element name="ViewParameter"
         type="typens:ViewParameter"/>
      <xsd:element name="Component" type="typens:Component"/>
      <xsd:element name="MaxHeight" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="CustomInputPara" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="DownloadEmbedded" type="xsd:boolean"
         default="false" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The ID of the object from which to retrieve the contents. Specify either the object ID or the object name and version number.

ViewParameter
The viewing parameters.

Component
The component from which to retrieve the contents. Specify either name, display name, or ID, and the value of the component. The following formats do not support specifying the component ID:

- ExcelData
- ExcelDisplay
- PDF
- RTF

To retrieve the entire report, specify 0.

MaxHeight
Required for a Reportlet. The maximum height, in points, based on the web page layout design. By default, MaxHeight is 0, which means there is no limit to the height of the Reportlet. In this case, the entire component is converted into a Reportlet.

CustomInputPara
The input parameters to send.

DownloadEmbedded
Specifies whether to send the attachment embedded in the body of the SOAP message or in a separate MIME boundary. If True, the response contains the attachment as embedded data. If False, the attachment is in a separate MIME boundary. The default value is False.

Response schema

```
<xsd:complexType name="GetContentResponse">
   <xsd:sequence>
      <xsd:element name="ContentRef" type="typens:Attachment"/>
      <xsd:element name="PostResponseRef" type="typens:Attachment"
         minOccurs="0"/>
      <xsd:element name="ComponentId" type="xsd:string"/>
      <xsd:element name="FileExtension" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="FileDescription" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ContentRef
Contains the following properties of the attachment:

- MimeType
- ContentEncoding
- ContentLength
- Locale

PostResponseRef
Used only when the request input format is Reportlet. Contains the Reportlet parameters.

ComponentId
The component ID.

FileExtension
The file extension.

FileDescription
The description of the file.

ConnectionHandle
The ID of the report. Supports viewing a report when the report is already in iHub System. Specified in the SOAP header.

GetCustomFormat

Invokes the Actuate Basic AcReport::GetCustomFormat method. Use GetCustomFormatData to extract the results of calling the AcReport::GetCustomFormat method. For example, if you implement code to create an Excel file, use GetCustomFormatData to retrieve the generated Excel file.

Request schema

```
<xsd:complexType name="GetCustomFormat">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
      <xsd:element name="ViewParameter"
         type="typens:ViewParameter"/>
      <xsd:element name="ArgumentList"
         type="typens:ArrayOfArgument"/>
      <xsd:element name="DownloadEmbedded" type="xsd:boolean"
         default="false" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The ID of the object from which to invoke the Actuate Basic AcReport:GetCustomFormat method.

ViewParameter
The viewing parameters.

ArgumentList
The list of the name and value pairs.

DownloadEmbedded
Specifies whether to send the attachment embedded in the body of the SOAP message or in a separate MIME boundary. If True, the response contains the attachment as embedded data. If False, the attachment is in a separate MIME boundary. The default value is False.

Response schema

```
<xsd:complexType name="GetCustomFormatResponse">
   <xsd:sequence>
      <xsd:element name="CustomRef" type="typens:Attachment"/>
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

CustomRef
The details of the attachment such as contentId, contentType, contentLength, contentEncoding, locale, and contentData.

ConnectionHandle
The ID of the report. Supports viewing a report when the report is already in iHub System. Specified in the SOAP header.

GetDatabaseConnectionDefinition

Retrieves information about an ACS database connection object.

Request schema

```
<xsd:complexType name="GetDatabaseConnectionDefinition">
   <xsd:sequence>
      <xsd:element name="FileName" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

FileName
The name of the database connection object for which to retrieve information.

Response schema

```
<xsd:complexType name="GetDatabaseConnectionDefinitionResponse">
   <xsd:sequence>
      <xsd:element name="DatabaseConnectionDefinition"
      type="typens:DatabaseConnectionDefinition"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

DatabaseConnectionDefinition
Information about the database connection object.

GetDatabaseConnectionParameters

Retrieves the connection parameter definitions that the database requires. Typically, you call GetDatabaseConnectionTypes to retrieve the list of available database types, then call GetDatabaseConnectionParameters to retrieve the connection parameters for a specific database type.

Request schema

```
<xsd:complexType name="GetDatabaseConnectionParameters">
   <xsd:sequence>
      <xsd:element name="Type" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Type
The type of database.

Response schema

```
<xsd:complexType name="GetDatabaseConnectionParametersResponse">
   <xsd:sequence>
      <xsd:element name="List"
         type="typens:ArrayOfParameterDefinition"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

List
Information about the connection parameters.

GetDatabaseConnectionTypes

Retrieves the list of available DBMS platforms. Table 8-2 lists the DBMS platforms that Actuate supports for Actuate Caching service (ACS) databases prior to iHub2.

Table 8-2 DBMS platforms that are supported for ACS databases

DBMS	Operating system
SQL Server 2005	Microsoft Windows
SQL Server 2008	Microsoft Windows

Request schema

```
<xsd:complexType name="GetDatabaseConnectionTypes"/>
```

Response schema

```
<xsd:complexType name="GetDatabaseConnectionTypesResponse">
   <xsd:sequence>
      <xsd:element name="List" type="typens:ArrayOfString"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

List
The list of available DBMS platforms.

Warnings
Any problems that occur.

GetDataExtractionFormats

Retrieves a list of DataExtractionFormat objects for a specific file type. These objects consist of an output format and a mime type.

Request schema

```
<xsd:complexType name="GetDataExtractionFormats">
   <xsd:sequence>
      <xsd:element name="FileType" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

FileType
The name of the file type about which to retrieve information.

Response schema

```
<xsd:complexType name="GetDataExtractionFormatsResponse">
   <xsd:sequence>
      <xsd:element name="DataExtractionFormats"
         type="typens:ArrayOfDataExtractionFormat"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

DataExtractionFormats
The list of DataExtractionFormat objects.

GetDocumentConversionOptions

Retrieves a list of DocumentConversionOptions. A document conversion option includes a file type, output format, mime type and parameter definitions.

Request schema

```
<xsd:complexType name="GetDocumentConversionOptions">
   <xsd:sequence>
      <xsd:element name="FileType" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="OutputFormat" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

FileType
The file type about which to retrieve the conversion options. When FileType is not set, the response includes options for all the Java report documents. In this case, GetDocumentConversionOptions ignores OutputFormat.

When a FileType is set, GetDocumentConversionOptions returns the options for the conversion to the specified OutputFormat.

OutputFormat
The display format about which to retrieve the conversion options. When OutputFormat is not specified, the response includes parameters for all output formats.

If the FileType or OutputFormat fields specify unsupported formats, GetDocumentConversionOptions returns a SOAP fault.

Response schema

```
<xsd:complexType name="GetDocumentConversionOptionsResponse">
   <xsd:sequence>
      <xsd:element name="ConversionOptions"
         type="typens:ArrayOfDocumentConversionOptions"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ConversionOptions
The list of DocumentConversionOptions

GetDynamicData

Retrieves dynamic data from a report.

Request schema

```
<xsd:complexType name="GetDynamicData">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
      <xsd:element name="ViewParameter"
         type="typens:ViewParameter" minOccurs="0"/>
      <xsd:element name="Component" type="typens:ComponentType"/>
      <xsd:element name="DownloadEmbedded" type="xsd:boolean"
         default="false" minOccurs="0"/>
      <xsd:element name="CoordinateX" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="CoordinateY" type="xsd:long"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The ID of the object from which to retrieve the dynamic data.

ViewParameter
The viewing parameters.

Component
The name, display name, or ID, and the value of the component for which to retrieve the dynamic data.

DownloadEmbedded
Specifies whether to send the attachment embedded in the body of the SOAP message or in a separate MIME boundary. If True, the response contains the attachment as embedded data. If False, the attachment is in a separate MIME boundary. The default value is False.

CoordinateX
The *x*-axis coordinate of the chart element the user chose. Required if Format is ImageMapURL.

CoordinateY
The *y*-axis coordinate of the chart element the user chose. Required if Format is ImageMapURL.

Response schema

```
<xsd:complexType name="GetDynamicDataResponse">
   <xsd:sequence>
      <xsd:element name="DynamicDataRef" type="typens:Attachment"/>
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="DataLinkingURL" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

DynamicDataRef
The details of the attachment such as contentId, contentType, contentLength, contentEncoding, locale, and contentData.

ConnectionHandle
The ID of the report. Supports viewing a report when the report is already in iHub System. Specified in the SOAP header.

DataLinkingURL
If Format is ImageMapURL, the URL of the hyperlink. If no hyperlink is associated with the URL, an empty string is returned. If DataLinkingURL is used, an attachment is not returned.

GetEmbeddedComponent

Retrieves an embedded component from a report.

Request schema

```
<xsd:complexType name="GetEmbeddedComponent">
   <xsd:sequence>
      <xsd:element name="ObjectId" type="xsd:string"/>
      <xsd:element name="Operation" type="xsd:string"/>
      <xsd:element name="StreamName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Embed" type="xsd:int" minOccurs="0"/>
```

```
      <xsd:element name="ComponentId" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ScalingFactor" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="AcceptEncoding" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="DownloadEmbedded" type="xsd:boolean"
         default="false" minOccurs="0"/>
      <xsd:element name="Format" type="xsd:string" minOccurs="0"/>
      <xsd:element name="CoordinateX" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="CoordinateY" type="xsd:long"\
         minOccurs="0"/>
      <xsd:element name="RedirectPath"type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ObjectId
The ID of the object from which to retrieve the data.

Operation
The type of data to retrieve. Valid values are:

- GetStaticData
  Retrieves static data
- GetDynamicData
  Retrieves dynamic data
- GetStyleSheet
  Retrieves the style sheet

StreamName
The stream name. Required if the operation is GetStaticData, optional otherwise.

Embed
Required if the operation is GetStaticData, optional otherwise.

ComponentId
The ID of the component from which to retrieve the data. Required if the operation is GetDynamicData, optional otherwise.

ScalingFactor
Supported only for GetDynamicData and GetStaticData operations. Adapts the size of a Reportlet to the Reportlet frame.

AcceptEncoding
The list of encoding methods the browser supports.

DownloadEmbedded
Specifies whether to send the attachment embedded in the body of the SOAP message or in a separate MIME boundary. If True, the response contains the attachment as embedded data. If False, the attachment is in a separate MIME boundary. The default value is False.

Format
Applies only if the operation is GetDynamicData. The format in which the report displays. To support users clicking a point in a chart to navigate to different report sections, set Format to ImageMapURL and set the CoordinateX and CoordinateY elements.

CoordinateX
The *x*-axis coordinate of the chart element the user chose. Required if Format is ImageMapURL.

CoordinateY
The *y*-axis coordinate of the chart element the user chose. Required if Format is ImageMapURL.

RedirectPath
The context path to the hyperlink.

Response schema

```
<xsd:complexType name="GetEmbeddedComponentResponse">
   <xsd:sequence>
      <xsd:element name="EmbeddedRef" type="typens:Attachment"/>
      <xsd:element name="EmbeddedRef" type="typens:Attachment"/>
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="DataLinkingURL" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

EmbeddedRef
Contains the following properties of the attachment:

- MimeType
- ContentEncoding
- ContentLength
- Locale

ConnectionHandle
The ID of the report. Supports viewing a report when the report is already in iHub System. Specified in the SOAP header.

DataLinkingURL
The URL of the hyperlink if Format is set to ImageMapURL. If no hyperlink is associated with the URL, an empty string is returned. If DataLinkingURL is used, no attachment is returned.

GetFactoryServiceInfo

Retrieves general information about a Factory service. The node name is specified in the TargetServer element of the SOAP header.

GetFactoryServiceInfo is available only to a BIRT iHub System administrator. To log in as a BIRT iHub System administrator, use SystemLogin.

To retrieve a list of servers for which you can obtain information, use GetSystemServerList.

Request schema

```
<xsd:complexType name="GetFactoryServiceInfo"/>
```

Request elements

GetFactoryServiceInfo
Information about the Factory service.

Response schema

```
<xsd:complexType name="GetFactoryServiceInfoResponse">
   <xsd:sequence>
      <xsd:element name="ServerName" type="xsd:string"/>
      <xsd:element name="PendingSyncJobs" type="xsd:long"/>
      <xsd:element name="SyncJobQueueSize" type="xsd:long"/>
      <xsd:element name="RunningSyncJobs" type="xsd:long"/>
      <xsd:element name="RunningJobs" type="xsd:long"/>
      <xsd:element name="SyncFactoryProcesses" type="xsd:long"/>
      <xsd:element name="MaxFactoryProcesses" type="xsd:long"/>
      <xsd:element name="TransientReportCacheSize"type="xsd:long"/>
      <xsd:element name="PercentTransientReportCacheInUse"
         type="xsd:long"/>
      <xsd:element name="CurrentTransientReportTimeout"
         type="xsd:long"/>
      <xsd:element name="TransientReportTimeout" type="xsd:long"/>
      <xsd:element name="SyncJobQueueWait" type="xsd:long"/>
      <xsd:element name="MaxSyncJobRuntime" type="xsd:long"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ServerName
The node for which information is returned.

PendingSyncJobs
The number of synchronous jobs in queue.

SyncJobQueueSize
The maximum number of synchronous jobs allowed in the queue.

RunningSyncJobs
The number of synchronous jobs currently running.

RunningJobs
The total number of jobs currently running.

SyncFactoryProcesses
The number of Factories reserved for running synchronous jobs.

MaxFactoryProcesses
The maximum number of Factories that can run on the system.

TransientReportCacheSize
The maximum disk space available for transient reports, in megabytes.

PercentTransientReportCacheInUse
The currently used percentage of disk space available for transient reports.

CurrentTransientReportTimeout
The number of minutes after which transient reports are deleted from the synchronous cache adjusted according to disk space currently available in the synchronous cache.

TransientReportTimeout
Time after which transient reports are deleted from the synchronous cache, in minutes.

SyncJobQueueWait
The maximum time a job remains in the synchronous queue, in seconds.

MaxSyncJobRuntime
The maximum job execution time, in seconds.

GetFactoryServiceJobs

Retrieves information about pending synchronous jobs and all running jobs on the node. The node is specified in the TargetServer element of the SOAP header.

For pending synchronous jobs, GetFactoryServiceJobs always returns the following information in addition to any values you specify:

- ConnectionHandle
- ObjectId

- IsTransient
- Volume

For running jobs, GetFactoryServiceJobs always returns the following information in addition to any values you specify:

- IsSyncJob
- Volume
- ConnectionHandle
- ObjectId
- IsTransient for synchronous jobs or JobId for asynchronous jobs

To retrieve all information, specify All.

GetFactoryServiceJobs is available only to a BIRT iHub System administrator. To log in as a BIRT iHub System administrator, use SystemLogin.

Request schema

```
<xsd:complexType name="GetFactoryServiceJobs">
   <xsd:sequence>
      <xsd:element name="PendingSyncJobsResultDef"
         type="typens:ArrayOfString" minOccurs="0"/>
      <xsd:element name="RunningJobsResultDef"
         type="typens:ArrayOfString" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

PendingSyncJobsResultDef
Requests the following information about pending synchronous jobs:

- Default
  ConnectionHandle, ObjectId, IsTransient, and Volume
- All
  All information
- ServerName
  The node on which the job originated
- Owner
  The name of the user who submitted the job
- ExecutableFileName
  The fully qualified name of the executable file
- ExecutableVersionNumber
  The fully qualified version number of the executable file

- ExecutableVersionName
  The fully qualified version name of the executable file
- SubmissionTime
  The time the job was submitted to the server
- QueueTimeout
  The number of seconds remaining before the job is deleted from the queue
- QueuePosition
  The job's position in the queue

RunningJobsResultDef
Requests the following information about all running jobs:

- Default
  IsSyncJob, Volume, ConnectionHandle, ObjectId, IsTransient for synchronous jobs or JobId for asynchronous jobs.
- All
  All information.
- ServerName
  The node on which the job originated.
- Owner
  The name of the user who submitted the job.
- ExecutableFileName
  The fully qualified name of the executable file.
- ExecutableVersionNumber
  The fully qualified version number of the executable file.
- ExecutableVersionName
  The fully qualified version name of the executable file.
- SubmissionTime
  The time the job was submitted to the server.
- StartTime
  The time at which the job execution started.
- ExecutionTimeout
  The number of seconds remaining before the job execution expires. Always zero (infinite) for asynchronous reports.

- IsSyncFactory
  True if the Factory is running synchronous jobs, False if the Factory is running asynchronous jobs.
- FactoryPid
  The Process ID of the Factory.

Response schema

```
<xsd:complexType name="GetFactoryServiceJobsResponse">
   <xsd:sequence>
      <xsd:element name="PendingSyncJobs"
         type="ArrayOfPendingSyncJob" minOccurs="0"/>
      <xsd:element name="RunningJobs"
         type="ArrayOfRunningJob" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

PendingSyncJobs
Information about pending synchronous jobs on the node.

RunningJobs
Information about all running jobs on the node.

GetFileACL

Retrieves the ACL of the specified Encyclopedia file or folder.

Request schema

```
<xsd:complexType name="GetFileACL">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="FileId" type="xsd:string"/>
         <xsd:element name="FileName" type="xsd:string">
         </xsd:element>
      </xsd:choice>
      <xsd:choice minOccurs="0">
         <xsd:element name="GrantedUserId" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="GrantedUserName" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="GrantedRoleId" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="GrantedRoleName" type="xsd:string"
            minOccurs="0"/>
      </xsd:choice>
         <xsd:element name="FetchSize" type="xsd:int"
            minOccurs="0"/>
         <xsd:element name="FetchDirection" type="xsd:boolean"
            minOccurs="0"/>
```

```
        <xsd:element name="CountLimit" type="xsd:int"
           minOccurs="0"/>
        <xsd:element name="FetchHandle" type="xsd:string"
           minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

FileId
The ID of the file or folder for which to retrieve the ACL. Specific either the FileId or the FileName.

FileName
The full name of the file or folder for which to retrieve the ACL. Specify either FileName or FileId.

GrantedUserId
The user ID.

GrantedUserName
The user name.

GrantedRoleId
The role ID.

GrantedRoleName
The role name.

FetchSize
The maximum number of records to retrieve and return in a result set. The default value is 500.

FetchDirection
If True, records are retrieved forward. If False, records are retrieved backward. The default value is True.

CountLimit
The maximum number of records to count. By default, CountLimit is equal to FetchSize. To count all records, set CountLimit to -1.

FetchHandle
Retrieves more items from the result set. In the second and subsequent calls for data, specify the same search criteria as in the original call.

Response schema

```
<xsd:complexType name="GetFileACLResponse">
   <xsd:sequence>
      <xsd:element name="ACL"type="typens:ArrayOfPermission"
         minOccurs="0"/>
      <xsd:element name="FetchHandle" type="xsd:string"
         minOccurs="0"/>
```

```
      <xsd:element name="TotalCount" type="xsd:int" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ACL
The ACL of the file or folder.

FetchHandle
Indicates that the number of items in the result set exceeds the FetchSize limit.

TotalCount
The number of entries in the search result set.

GetFileCreationACL

Retrieves the specified user's FileCreationACL. The FileCreationACL is the template applied to all new files the user creates.

Request schema

```
<xsd:complexType name="GetFileCreationACL">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="CreatedByUserName" type="xsd:string"/>
         <xsd:element name="CreatedByUserId" type="xsd:string"/>
      </xsd:choice>
      <xsd:choice minOccurs="0">
         <xsd:element name="GrantedUserId" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="GrantedUserName" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="GrantedRoleId" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="GrantedRoleName" type="xsd:string"
            minOccurs="0"/>
      </xsd:choice>
      <xsd:element name="FetchSize" type="xsd:int" minOccurs="0"/>
      <xsd:element name="FetchDirection" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="CountLimit" type="xsd:int" minOccurs="0"/>
      <xsd:element name="FetchHandle" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

CreatedByUserName
The name of the user whose template to retrieve. Specify either CreatedByUserName or CreatedByUserId.

CreatedByUserId
The ID of the user whose template to retrieve. Specify either CreatedByUserId or CreatedByUserName.

GrantedUserId
The user ID. Specify either GrantedUserId or GrantedUserName.

GrantedUserName
The user name. Specify either GrantedUserName or GrantedUserId.

GrantedRoleId
The role ID. Specify either GrantedRoleId or GrantedRoleName.

GrantedRoleName
The role name. Specify either GrantedRoleName or GrantedRoleId.

FetchSize
The maximum number of records to retrieve and return in a result set. The default value is 500.

FetchDirection
If True, records are retrieved forward. If False, records are retrieved backward. The default value is True.

CountLimit
The maximum number of records to count. By default, CountLimit is equal to FetchSize. To count all records, set CountLimit to -1.

FetchHandle
Retrieves more items from the result set. In the second and subsequent calls for data, specify the same search criteria as in the original call.

Response schema

```
<xsd:complexType name="GetFileCreationACLResponse">
   <xsd:sequence>
      <xsd:element name="ACL" type="typens:ArrayOfPermission"
         minOccurs="0"/>
      <xsd:element name="FetchHandle" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="TotalCount" type="xsd:int" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ACL
The user's ACL.

FetchHandle
Indicates that the number of items in the result set exceeds the FetchSize limit.

TotalCount
The number of entries in the search result set.

GetFileDetails

Retrieves the properties of the specified file.

Request schema

```
<xsd:complexType name="GetFileDetails">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="FileName" type="xsd:string"/>
         <xsd:element name="FileId" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="ResultDef" type="typens:ArrayOfString">
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

FileName
The full name of the file for which to retrieve properties. Specify either FileName or FileId.

FileId
The ID of the file for which to retrieve properties. Specify either FileId or FileName.

ResultDef
The properties to retrieve. The file properties are always returned. In addition, you can specify one or more of the following:

- ACL
  The access control list (ACL) of the file
- ArchiveRules
  The archive rules of the file
- AccessType
  The access rights to the file, private or shared

Response schema

```
<xsd:complexType name="GetFileDetailsResponse">
   <xsd:sequence>
      <xsd:element name="File" type="typens:File"/>
      <xsd:element name="ACL" type="typens:ArrayOfPermission"
         minOccurs="0"/>
      <xsd:element name="ArchiveRules"
         type="typens:ArrayOfArchiveRule"minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

File
The file properties.

ACL
The ACL. Returned only if ACL is specified in ResultDef.

ArchiveRules
The archive rules. Returned only if ArchiveRules is specified in ResultDef.

GetFileTypeParameterDefinitions

Retrieves parameters of the specified file type on the BIRT iHub to which the user is logged in.

Request schema

```
<xsd:complexType name="GetFileTypeParameterDefinitions">
   <xsd:sequence>
      <xsd:element name="FileType" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

FileType
The name of the file type for which to retrieve information.

Response schema

```
<xsd:complexType name="GetFileTypeParameterDefinitionsResponse">
   <xsd:sequence>
      <xsd:element name="ParameterList"
         type="typens:ArrayOfParameterDefinition"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ParameterList
The list of parameters.

GetFolderItems

Retrieves all specified objects in a specified folder, such as all files or folders, a list of files or folders, or all users.

To search all files or folders that match the specified conditions, specify Search.

Request schema

```
<xsd:complexType name="GetFolderItems">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="FolderName" type="xsd:string">
         </xsd:element>
         <xsd:element name="FolderId" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="ResultDef" type="typens:ArrayOfString">
      </xsd:element>
      <xsd:element name="LatestVersionOnly" type="xsd:boolean"
```

```
        minOccurs="0"/>
      <xsd:element name="Search" type="typens:FileSearch"
        minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

FolderName
The full path and the name of the folder from which to retrieve objects. Specify either FolderName or FolderId.

FolderId
The ID of the folder from which to retrieve objects. Specify either FolderId or FolderName.

ResultDef
The properties to retrieve. By default, the Id and Name are always returned. In addition, you can specify the following properties:

- Description
- FileType
- Owner
- PageCount
- Size
- TimeStamp
- Version
- VersionName
- UserPermissions

LatestVersionOnly
Specifies whether only the latest version is returned. If True, only the latest version is returned. The default value is False.

Search
The search condition. If conditions apply to multiple fields, use ConditionArray.

Response schema

```
<xsd:complexType name="GetFolderItemsResponse">
   <xsd:sequence>
      <xsd:element name="ItemList" type="typens:ArrayOfFile"/>
      <xsd:element name="FetchHandle" type="xsd:string"
        minOccurs="0"/>
      <xsd:element name="TotalCount" type="xsd:int" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ItemList
The objects matching the search criteria.

FetchHandle
Indicates that the number of items in the result set exceeds the FetchSize limit.

TotalCount
The number of entries in the search result set.

GetFormats

Retrieves a list of locales and formats the server supports.

Request schema

```
<xsd:complexType name="GetFormats">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
      <xsd:element name="FormatType" type="typens:FormatType"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The ID of the object from which to retrieve information.

FormatType
One of the following formats to return:

- 0
  All formats
- 1
  View formats
- 2
  Search formats

If you do not specify a format, all formats are returned.

Response schema

```
<xsd:complexType name="GetFormatsResponse">
   <xsd:sequence>
      <xsd:element name="FormatList" type="typens:ArrayOfString"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

FormatList
The list of formats.

GetInfoObject

Retrieves an information object.

Request schema

```
<xsd:complexType name="GetInfoObject">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="InfoObjectName" type="xsd:string" />
         <xsd:element name="InfoObjectId" type="xsd:string" />
         <xsd:element name="SupportedQueryFeatures"
            type="typens:SupportedQueryFeatures" minOccurs="0" />
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

InfoObjectName
The name of the information object.

InfoObjectId
The object ID of the information object.

SupportedQueryFeatures
Other features on which to query the information object.

Response schema

```
<xsd:complexType name="GetInfoObjectResponse">
   <xsd:sequence>
      <xsd:element name="InfoObject" type="typens:Query" />
   </xsd:sequence>
</xsd:complexType>
```

Response elements

InfoObject
The information object.

GetJavaReportEmbeddedComponent

Retrieves an embedded component in the document such as an image or a graph.

Request schema

```
<xsd:complexType name="GetJavaReportEmbeddedComponent">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
      <xsd:element name="Component"
         type="typens:ArrayOfNameValuePair"/>
      <xsd:element name="Attributes"
         type="typens:ArrayOfNameValuePair" minOccurs="0"/>
      <xsd:element name="DownloadEmbedded" type="xsd:boolean"
            default="false" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The ID of the object from which to retrieve an embedded component in a document.

Component
The name, display name, or ID, and the value of the component to retrieve.

Attributes
Currently not used by BIRT reports.

DownloadEmbedded
Specifies whether to send the attachment embedded in the body of the SOAP message or in a separate MIME boundary. If True, the response contains the attachment as embedded data. If False, the attachment is in a separate MIME boundary. The default value is False.

Response schema

```
<xsd:complexType name="GetJavaReportEmbededComponentResponse">
   <xsd:sequence>
      <xsd:element name="EmbeddedRef" type="typens:Attachment"/>
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

EmbeddedRef
Contains the following properties of the attachment:

- MimeType
- ContentEncoding
- ContentLength
- Locale

ConnectionHandle
The ID of the report. Supports viewing a report when the report is already in iHub System. Specified in the SOAP header.

GetJavaReportTOC

Retrieves the table of contents (TOC) of the document.

Request schema

```
<xsd:complexType name="GetJavaReportTOC">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
      <xsd:element name="DownloadEmbedded" type="xsd:boolean"
         default="false" minOccurs="0"/>
      <xsd:element name="Component"
         type="typens:ArrayOfNameValuePair" minOccurs="0"/>
      <xsd:element name="Recursive" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The ID of the object from which to retrieve the TOC.

DownloadEmbedded
Specifies whether to send the attachment embedded in the body of the SOAP message or in a separate MIME boundary. If True, the response contains the attachment as embedded data. If False, the attachment is in a separate MIME boundary. The default value is False.

Component
The name, display name, or ID, and the value of the component from which to retrieve the TOC.

Recursive
Specifies whether to search subfolders. If True, the search includes subfolders. The default value is False.

Response schema

```
<xsd:complexType name="GetJavaReportTOCResponse">
   <xsd:sequence>
      <xsd:element name="TOCRef" type="typens:Attachment"
         minOccurs="0"/>
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

TocRef
The details of the attachment such as contentId, contentType, contentLength, contentEncoding, locale, and contentData.

ConnectionHandle
The ID of the report. Supports viewing a report when the report is already in iHub System. Specified in the SOAP header.

GetJobDetails

Retrieves the properties of a specified job.

Request schema

```
<xsd:complexType name="GetJobDetails">
   <xsd:sequence>
      <xsd:element name="JobId" type="xsd:string"/>
      <xsd:element name="ResultDef" type="typens:ArrayOfString"/>
      <xsd:element name="GroupingEnabled" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="SupportedQueryFeatures"
         type="typens:SupportedQueryFeatures" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

JobId
The ID of the job from which to retrieve properties.

ResultDef
The properties to retrieve. You can specify the following properties:

- JobAttributes
  The general job properties
- InputDetail
  The job input parameters
- Schedules
  The job schedule information
- PrinterOptions
  The printer settings, if available
- NotifyUsers
  The names of users to receive notifications about the job
- NotifyGroups
  The names of groups to receive notifications about the job
- NotifyChannels
  The names of channels to receive notifications about the job
- DefaultOutputFileACL
  The output file ACL templates
- Status
  The job status
- ReportParameters
  The report parameters from the report parameters value file associated with the job
- ResourceGroup
  The name of the resource group to which the job is assigned

GroupingEnabled
Provided for backward compatibility. If the DOX was created using Actuate 7 Service Pack 2 or a later release, set GroupingEnabled to True. In this case, BIRT iHub Information and Management Console enable the grouping and aggregation pages. If the DOX came from an earlier version of an Actuate product, set GroupingEnabled to False.

SupportedQueryFeatures
Specifies additional query features.

Response schema

```
<xsd:complexType name="GetJobDetailsResponse">
   <xsd:sequence>
      <xsd:element name="JobAttributes" type="typens:JobProperties"
         minOccurs="0"/>
      <xsd:element name="InputDetail" type="typens:JobInputDetail"
         minOccurs="0"/>
      <xsd:element name="Schedules" type="typens:JobSchedule"
         minOccurs="0"/>
      <xsd:element name="PrinterOptions"
         type="typens:PrinterOptions" minOccurs="0"/>
      <xsd:element name="NotifyUsers" type="typens:ArrayOfString"
         minOccurs="0"/>
      <xsd:element name="NotifyGroups"
         type="typens:ArrayOfString">
      <xsd:element name="NotifyChannels"
         type="typens:ArrayOfString" minOccurs="0"/>
      <xsd:element name="DefaultOutputFileACL"
         type="typens:ArrayOfPermission" minOccurs="0"/>
      <xsd:element name="Status" type="xsd:string" minOccurs="0"/>
      <xsd:element name="ReportParameters"
         type="typens:ArrayOfParameterValue" minOccurs="0"/>
      <xsd:element name="Query" type="typens:Query"
         minOccurs="0"/>
      <xsd:element name="OutputFileAccessType"
         type="typens:FileAccess" minOccurs="0"/>
      <xsd:element name="WaitForEvent" type="typens:Event"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

JobAttributes
The general job attributes.

InputDetail
The job input parameters.

Schedules
The job schedule information.

PrinterOptions
The job printer settings.

NotifyUsers
The names of users to receive notifications about the job.

NotifyGroups
The names of groups to receive notifications about the job.

NotifyChannels
The names of channels to receive notifications about the job.

DefaultOutputFileACL
The output file access control list (ACL) templates.

Status
The job status.

ReportParameters
The report parameters from the report object value (.rov) file associated with the job.

Query
The data object values (.dov) file associated with the job.

OutputFileAccessType
The access rights to the output file. One of the following values:

- Private
  Only the owner of the file and an administrator can access the file.
- Shared
  All users and roles specified in the access control list (ACL) for the file can access the file.

WaitForEvent
An event that must complete before processing the response.

GetMetaData

Retrieves the metadata describing a result set schema.

Request schema

```
<xsd:complexType name="GetMetaData">
   xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
      <xsd:element name="Component"
         type="typens:ArrayOfNameValuePair" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The object from which to retrieve the metadata.

Component
The name, display name, or ID, and the value of the component from which to retrieve the metadata.

Response schema

```
<xsd:complexType name="GetMetaDataResponse">
   <xsd:sequence>
      <xsd:element name="ArrayOfResultSetSchema"
         type="typens:ArrayOfResultSetSchema" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ArrayOfResultSetSchemat
The complex data type that represents an array of ResultSetSchema objects.

GetNoticeJobDetails

Retrieves the properties of the specified job notice.

Request schema

```
<xsd:complexType name="GetNoticeJobDetails">
   <xsd:sequence>
      <xsd:element name="JobId" type="xsd:string"/>
      <xsd:element name="ResultDef" type="typens:ArrayOfString"/>
      <xsd:element name="NotifiedChannelId" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="NotifiedChannelName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="GroupingEnabled" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="SupportedQueryFeatures"
         type="typens:SupportedQueryFeatures" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

JobId
The ID of the job from which to retrieve properties.

ResultDef
The properties to retrieve. You can specify the following properties:

- InputDetail
  The job input parameters
- Schedules
  The job schedule information
- PrinterOptions
  The printer settings, if available
- NotifyUsers
  The names of users to receive notifications about the job

- NotifyGroups
  The names of groups to receive notifications about the job
- NotifyChannels
  The names of channels to receive notifications about the job
- DefaultOutputFileACL
  The output file access control list (ACL) templates
- Status
  The job status
- ReportParameters
  The report parameters from the report object value (.rov) file associated with the job
- ResourceGroup
  The name of the resource group to which the job is assigned

NotifiedChannelId
The ID of the channel which received the notice.

NotifiedChannelName
The name of the channel which received the notice.

GroupingEnabled
Provided for backward compatibility. If the Actuate Basic information object executable (.dox) file was created using Actuate 7 Service Pack 2 or a later release, set GroupingEnabled to True. In this case, BIRT iHub Information and Management Consoles enable the grouping and aggregation pages. If the DOX was created using an earlier version of an Actuate product, set GroupingEnabled to False.

SupportedQueryFeatures
Specifies additional query features.

Response schema

```
<xsd:complexType name="GetNoticeJobDetailsResponse">
   <xsd:sequence>
      <xsd:element name="JobAttributes"
         type="typens:JobProperties"/>
      <xsd:element name="InputDetail" type="typens:JobInputDetail"
         minOccurs="0"/>
      <xsd:element name="Schedules" type="typens:JobSchedule"
         minOccurs="0"/>
      <xsd:element name="PrinterOptions"
         type="typens:JobPrinterOptions" minOccurs="0"/>
      <xsd:element name="NotifyUsers" type="typens:ArrayOfString"
         minOccurs="0"/>
```

```
      <xsd:element name="NotifyGroups" type="typens:ArrayOfString"
         minOccurs="0"/>
      <xsd:element name="NotifyChannels"
         type="typens:ArrayOfString" minOccurs="0"/>
      <xsd:element name="DefaultOutputFileACL"
         type="typens:ArrayOfPermission" minOccurs="0"/>
      <xsd:element name="Status" type="xsd:string" minOccurs="0"/>
      <xsd:element name="ReportParameters"
         type="typens:ArrayOfParameterValue" minOccurs="0"/>
      <xsd:element name="Query" type="typens:Query" minOccurs="0"/>
      <xsd:element name="OutputFileAccessType"
         type="typens:FileAccess" minOccurs="0"/>
      <xsd:element name="WaitForEvent" type="typens:Event"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

JobAttributes
The general job attributes.

InputDetail
The job input parameters.

Schedules
The job schedule information.

PrinterOptions
The job printer settings.

NotifyUsers
The names of users to receive notifications about the job.

NotifyGroups
The names of groups to receive notifications about the job.

NotifyChannels
The names of channels to receive notifications about the job.

DefaultOutputFileACL
The output file access control list (ACL) templates.

Status
The job status.

ReportParameters
The report parameters from report object value (.rov) file associated with the job.

Query
The data object values (.dov) file associated with the job.

OutputFileAccessType
The access rights to the output file. One of the following values:

- Private
  Only the owner of the file and an administrator can access the file.
- Shared
  All users and roles specified in the ACL for the file can access the file.

WaitForEvent
An event that must be completed before the response is processed.

GetPageCount

Retrieves the number of pages in a report.

Request schema

```
<xsd:complexType name="GetPageCount">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The object from which to retrieve page count.

Response schema

```
<xsd:complexType name="GetPageCountResponse">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="PageCount" type="xsd:string"/>
         <xsd:element name="IsReportCompleted" type="xsd:boolean"/>
         <xsd:element name="ConnectionHandle" type="xsd:string"
            minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

Response elements

PageCount
The number of pages.

IsReportCompleted
True if report generation is complete, False otherwise.

ConnectionHandle
The ID of the report. Supports viewing a report when the report is already in iHub System. Specified in the SOAP header.

GetPageNumber

Retrieves the page number of a bookmark component in a report.

Request schema

```
<xsd:complexType name="GetPageNumber">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
      <xsd:element name="Component"
         type="typens:ArrayOfNameValuePair"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The object from which to retrieve the page number.

Component
The name, display name, or ID, and the value of the component from which to retrieve the page number. The page number is a bookmark value in a report.

Response schema

```
<xsd:complexType name="GetPageNumberResponse">
   <xsd:sequence>
      <xsd:element name="PageNumber" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

PageNumber
The component element specifies ArrayOfNameValuePair as the data type. If multiple valid bookmarks are present inside a report, a response only returns the page number for first bookmark. Actuate IDAPI does not support getting the page number for multiple components. If you specify multiple components, the response only returns the page number for the first component.

GetParameterPickList

Retrieves the parameters names from a pick list in a report.

Request schema

```
<xsd:complexType name="GetParameterPickList">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"
         minOccurs="0"/>
      <xsd:element name="CascadingGroupName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ParameterName" type="xsd:string"/>
      <xsd:element name="PrecedingParameterValues"
         type="typens:ArrayOfParameterValue" minOccurs="0"/>
      <xsd:element name="Filter" type="xsd:string" minOccurs="0"/>
```

```
      <xsd:element name="StartIndex" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="FetchSize" type="xsd:long" minOccurs="0"/>
      <xsd:element name="CountLimit" type="xsd:long"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The identifier of the object from which to retrieve the parameter pick list.

CascadingGroupName
The cascading group name in the pick list containing the target list.

ParameterName
The name of the parameter. Each parameter name is unique within a report even between execution and view parameters.

PrecedingParameterValues
The values of the parameters that precede the specified parameter.

Filter
A string prefix to be applied to the overall selection list.

StartIndex
The index where the fetch operation starts.

FetchSize
The maximum number of records to retrieve and return in a result set. The default value is 500.

CountLimit
The Number of entries to be counted after FetchSize is reached. By default, CountLimit is equal to FetchSize. To count all records, set CountLimit to -1.

Response schema

```
<xsd:complexType name="GetParameterPickListResponse">
   <xsd:sequence>
      <xsd:element name="ParameterPickList"
         type="typens:ArrayOfNameValuePair"/>
      <xsd:element name="TotalCount" type="xsd:long"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ParameterPickList
List of parameters.

TotalCount
The number of parameters in the list.

GetReportParameters

Retrieves report parameter values. GetReportParameters requires execute or read privileges when executed against a BIRT design, BIRT data design, or BIRT 360 dashboard file. When executed against a BIRT document file, GetReportParameters requires execute, secure read, or read privileges.

Request schema

```
<xsd:complexType name="GetReportParameters">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="JobId" type="xsd:string"/>
         <xsd:element name="ReportFileId" type="xsd:string"/>
         <xsd:element name="ReportFileName" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="ReportParameterType"
         type="typens:ReportParameterType" minOccurs="0"/>
      <xsd:element name="WithoutDynamicPickList"
         type="xsd:boolean" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

JobId
The ID of the job from which to retrieve the parameter values. Specify JobId to retrieve the parameter values from the report file associated with the job.

ReportFileId
The ID of the file from which to retrieve the parameter values. Specify ReportFileId or Report FileName to retrieve parameter values from an executable, document, or object value file, or a third-party compound storage file.

ReportFileName
The name of the file from which to retrieve the parameter values. Specify ReportFileName or Report FileId to retrieve parameter values from an executable, document, or object value file, or a third-party compound storage file.

ReportParameterType
Optional parameter type can include Execution, View, and All. If not specified, only execution parameters return to maintain backward compatibility.

WithoutDynamicPickList
If set to True, parameters will be returned without a dynamic pick list.

Response schema

```
<xsd:complexType name="GetReportParametersResponse">
   <xsd:sequence>
      <xsd:element name="ParameterList"
         type="typens:ArrayOfParameterDefinition"/>
```

```
      <xsd:element name="ViewParameterList"
         type="typens:ArrayOfParameterDefinition" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ParameterList
The list of parameter definition values such as group, cascading parent name, name, data type, default value, display name, help text, and so forth.

ViewParameterList
The list of view parameter definition values such as format, user agent, scaling factor, accept encoding, view operation, path information, embedded object path, redirect path, and PDF quality.

GetResourceGroupInfo

Retrieves information about a specific resource group.

Request schema

```
<xsd:complexType name="GetResourceGroupInfo">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Name
The name of the resource group for which to retrieve information.

Response schema

```
<xsd:complexType name="GetResourceGroupInfoResponse">
   <xsd:sequence>
      <xsd:element name="ResourceGroup"
         type="typens:ResourceGroup"/>
      <xsd:element name="ResourceGroupSettingsList"
         type="typens:ArrayOfResourceGroupSettings"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ResourceGroup
Contains the following information about the resource group:

- Name
- Disabled
- Description
- Type
- MinPriority
  Applies only to an asynchronous resource group

- MaxPriority
  Applies only to an asynchronous resource group
- Reserved
  Applies only to a synchronous resource group

ResourceGroupSettingsList
Contains the following information about the resource group:

- ServerName
- Activate
- MaxFactory
- FileTypes

GetResourceGroupList

Retrieves a list of resource groups available to a BIRT iHub. GetResourceGroupList returns two lists, one for asynchronous resource groups and one for synchronous resource groups.

Request schema

```
<xsd:complexType name="GetResourceGroupList">
   <xsd:sequence/>
</xsd:complexType>
```

Response schema

```
<xsd:complexType name="GetResourceGroupListResponse">
   <xsd:sequence>
      <xsd:element name="AsyncResourceGroupList"
         type="typens:ArrayOfResourceGroup"/>
      <xsd:element name="SyncResourceGroupList"
         type="typens:ArrayOfResourceGroup"/>
      <xsd:element name="ViewResourceGroupList"
         type="typens:ArrayOfResourceGroup"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

AsyncResourceGroupList
The list of available asynchronous resource groups and the properties of each of those resource groups.

SyncResourceGroupList
The list of available synchronous resource groups and the properties of each of those resource groups.

ViewResourceGroupList
The list of available synchronous resource groups and the properties of each of those resource groups.

GetSavedSearch

Retrieves a saved search.

Request schema

```
<xsd:complexType name="GetSavedSearch">
   <xsd:sequence>
      <xsd:element name="SearchObject"
         type="typens:ObjectIdentifier" />
      <xsd:element name="BasedOnObject"
         type="typens:ObjectIdentifier" />
   </xsd:sequence>
</xsd:complexType>
```

Request elements

SearchObject
The item a search was created on.

BasedOnObject
The object within the search object to search.

Response schema

```
<xsd:complexType name="GetSavedSearchResponse">
   <xsd:sequence>
      <xsd:element name="SelectList"
         type="typens:ArrayOfComponentIdentifier" />
      <xsd:element name="SearchList" type="typens:ArrayOfComponent"
         minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

SelectList
The list of where to search within an object.

SearchList
The items to search for.

GetServerResourceGroupConfiguration

Retrieves information about resource groups available to the specified BIRT iHub.

Request schema

```
<xsd:complexType name="GetServerResourceGroupConfiguration">
   <xsd:sequence>
      <xsd:element name="ServerName" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ServerName
The name of the BIRT iHub from which to retrieve information.

Response schema

```
<xsd:complexType
     name="GetServerResourceGroupConfigurationResponse">
   <xsd:sequence>
      <xsd:element name="ServerResourceGroupSettingList"
         type="typens:ArrayOfServerResourceGroupSetting"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ServerResourceGroupSettingList
Contains the following information about each resource group:

- ResourceGroupName
- Activate
- Type
- MaxFactory
- FileTypes

GetStaticData

Retrieves static types of data from a report. Static data is information within a report that does not change during the run of the report, such as images or other resources.

Request schema

```
<xsd:complexType name="GetStaticData">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
      <xsd:element name="ViewParameter"
         type="typens:ViewParameter" minOccurs="0"/>
      <xsd:element name="Stream" type="typens:Stream"/>
      <xsd:element name="DownloadEmbedded" type="xsd:boolean"
         default="false" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The ID of the object from which to retrieve static data.

ViewParameter
The viewing parameters.

Stream
The stream name and embedded property.

DownloadEmbedded
Specifies whether to send the attachment embedded in the body of the SOAP message or in a separate MIME boundary. If True, the response contains the

attachment as embedded data. If False, the attachment is in a separate MIME boundary. The default value is False.

Response schema

```
<xsd:complexType name="GetStaticDataResponse">
   <xsd:sequence>
      <xsd:element name="StaticDataRef" type="typens:Attachment"/>
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

StaticDataRef
The details of the attachment.

ConnectionHandle
The ID of the report. Supports viewing a report when the report is already in iHub System. Specified in the SOAP header.

GetStyleSheet

Retrieves the style sheet from a report.

Request schema

```
<xsd:complexType name="GetStyleSheet">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
      <xsd:element name="ViewParameter"
         type="typens:ViewParameter"/>
      <xsd:element name="DownloadEmbedded" type="xsd:boolean"
         default="false" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The ID of the object from which to retrieve the style sheet.

ViewParameter
The viewing parameters.

DownloadEmbedded
Specifies whether to send the attachment embedded in the body of the SOAP message or in a separate MIME boundary. If True, the response contains the attachment as embedded data. If False, the attachment is in a separate MIME boundary. The default value is False.

Response schema

```
<xsd:complexType name="GetStyleSheetResponse">
   <xsd:sequence>
      <xsd:element name="StyleSheetRef" type="typens:Attachment"/>
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

StyleSheetRef
The details of the attachment.

ConnectionHandle
The ID of the report. Supports viewing a report when the report is already in iHub System. Specified in the SOAP header.

GetSyncJobInfo

Retrieves information about a synchronous job.

GetSyncJobInfo is available only to a BIRT iHub System administrator. To log in as a BIRT iHub System administrator, use SystemLogin.

Request schema

```
<xsd:complexType name="GetSyncJobInfo">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="ObjectId" type="xsd:string">
         </xsd:element>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

Request elements

ObjectId
The ID of the synchronous job for which to retrieve information.

Response schema

```
<xsd:complexType name="GetSyncJobInfoResponse">
   <xsd:sequence>
      <xsd:element name="Status">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="Pending"/>
               <xsd:enumeration value="Running"/>
               <xsd:enumeration value="Completed"/>
               <xsd:enumeration value="Failed"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="ErrorDescription" type="xsd:string"
         minOccurs="0"/>
```

```
      <xsd:element name="Pending" type="PendingSyncJob"
         minOccurs="0"/>
      <xsd:element name="Running" type="RunningJob" minOccurs="0"/
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Status
The status of the job. Valid values are:

- Pending
- Running
- Completed
- Failed

ErrorDescription
The description of the error. Returned if Status is Failed.

Pending
The properties of the pending synchronous job.

Running
The properties of the running synchronous job.

GetSystemMDSInfo

Retrieves the names and properties of an MDS in a cluster or stand-alone server without authenticating the client. Use GetSystemMDSInfo to route requests to an alternate MDS if the one to which the client connects fails.

Request schema

```
<xsd:complexType name="GetSystemMDSInfo">
   <xsd:sequence>
      <xsd:element name="OnlineOnly" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

OnlineOnly
If True, information is retrieved only for an online MDS.

Response schema

```
<xsd:complexType name="GetSystemMDSInfoResponse">
   <xsd:sequence>
      <xsd:element name="MDSInfoList"
         type="typens:ArrayOfMDSInfo">
   </xsd:sequence>
</xsd:complexType>
```

Response elements

MDSInfoList
The information about each MDS.

GetSystemPrinters

Retrieves all system printer information on the BIRT iHub to which the user is logged in.

If GetSystemPrinters cannot retrieve the printer information, for example if the printer is temporarily unavailable, and you specified the PrinterName, the response contains empty or incomplete attributes. If GetSystemPrinters cannot retrieve the printer information and you did not specify PrinterName, a SOAP fault is generated.

Request schema

```
<xsd:complexType name="GetSystemPrinters">
   <xsd:sequence>
      <xsd:element name="PrinterName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="GetAllPaperSizes" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

PrinterName
The name of the printer for which to retrieve information. If not specified, information is retrieved for all printers configured for the volume.

GetAllPaperSizes
Indicates whether to retrieve all available paper sizes for the printer. If True, all paper sizes are retrieved. If False, only a subset of paper sizes are retrieved. The default value is False.

Response schema

```
<xsd:complexType name="GetSystemPrintersResponse">
   <xsd:sequence>
      <xsd:element name="Printers" type="typens:ArrayOfPrinter"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Printers
The printer information.

GetSystemServerList

Retrieves the list of BIRT iHubs and their states.

GetSystemServerList can retrieve information about cluster servers and online stand-alone servers. GetSystemServerList cannot retrieve information about stand-alone servers that are offline.

GetSystemServerList is available only to a BIRT iHub System administrator. To log in as a BIRT iHub System administrator, use SystemLogin.

Request schema

```
<xsd:complexType name="GetSystemServerList"/>
```

Response schema

```
<xsd:complexType name="GetSystemServerListResponse">
   <xsd:sequence>
      <xsd:element name="ServerList"
         type="typens:ArrayOfServerInformation"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ServerList
Contains the following information about the server:

- Server name
- Server state
- Error code of a failed server
- Error description of a failed server

GetSystemVolumeNames

Retrieves the names of volumes in a stand-alone server or cluster system without authentication. You can retrieve the names of all volumes or only online volumes.

Use GetSystemVolumeNames to populate a Login page with volume names.

Request schema

```
<xsd:complexType name="GetSystemVolumeNames">
   <xsd:sequence>
      <xsd:element name="OnlineOnly" type="xsd:boolean"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

OnlineOnly
Specifies whether names of all volumes or only online volumes are retrieved. If True, only names of online volumes are retrieved.

Response schema

```
<xsd:complexType name="GetSystemVolumeNamesResponse">
   <xsd:sequence>
      <xsd:element name="SystemName" type="xsd:string"/>
      <xsd:element name="SystemRestart" type="xsd:boolean"/>
      <xsd:element name="VolumeNameList"
         type="typens:ArrayOfString"/>
      <xsd:element name="SystemDefaultVolume" type="xsd:string"/>
      <xsd:element name="ServerVersionInformation"
         type="typens:ServerVersionInformation"/>
      <xsd:element name="ExpirationDate" type="xsd:string"/>
      <xsd:element name="DaysToExpiration" type="xsd:string"
         minOccurs="0"/
```

```
      <xsd:element name="NodeLockViolation" type="xsd:boolean"
        minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

SystemName
The name of the system.

SystemRestart
Specifies whether a system restart is required.

VolumeNameList
The list of volume names.

SystemDefaultVolume
The name of the default volume on a cluster server. Returns an empty value if no volume is specified as the default volume or if OnlineOnly is set to True and the default volume is offline.

ServerVersionInformation
Contains the following information about the server version:

- ServerVersion
- ServerBuild
- OSVersion

ExpirationDate
The expiration date. Returns NONE if the system license key does not expire.

DaysToExpiration
The number of days remaining until the System License Key expires.

NodeLockViolation
Specifies whether a licensing node-lock violation exists. The default value is False.

GetTOC

Returns the report's table of contents in XMLDisplay format.

Request schema

```
<xsd:complexType name="GetTOC">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
      <xsd:element name="ViewParameter"
        type="typens:ViewParameter"/>
      <xsd:element name="TocNodeId" type="xsd:string"/>
      <xsd:element name="Depth" type="xsd:string"/>
      <xsd:element name="DownloadEmbedded" type="xsd:boolean"
```

```
        default="false"
    </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The ID of the object from which to retrieve the table of contents.

ViewParameter
The viewing parameters. To support users clicking a point in a chart to navigate to different report sections, set the ViewParameter Format to ImageMapURL.

TocNodeId
The ID of the table of contents node.

Depth
The depth of the table of contents.

DownloadEmbedded
Specifies whether to send the attachment embedded in the body of the SOAP message or in a separate MIME boundary. If True, the response contains the attachment as embedded data. If False, the attachment is in a separate MIME boundary. The default value is False.

Response schema

```
<xsd:complexType name="GetTOCResponse">
   <xsd:sequence>
      <xsd:element name="TocRef" type="typens:Attachment"/>
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

TocRef
The details of the attachment such as contentId, contentType, contentLength, contentEncoding, locale, and contentData.

ConnectionHandle
The ID of the report. Supports viewing a report when the report is already in iHub System. Specified in the SOAP header.

GetUserLicenseOptions

Retrieves the specified user's license options.

Request schema

```
<xsd:complexType name="GetUserLicenseOptions">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="UserId" type="xsd:string"/>
         <xsd:element name="UserName" type="xsd:string"/>
      </xsd:choice>
```

```
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UserId
The ID of the user for whom to retrieve the printer settings. Specify either UserId or UserName.

UserName
The name of the user for whom to retrieve the printer settings. Specify either UserName or UserId.

Response schema

```
<xsd:complexType name="GetUserLicenseOptionsResponse">
   <xsd:sequence>
      <xsd:element name="LicenseOptions"
         type="typens:ArrayOfString"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

LicenseOptions
The license options.

GetUserPrinterOptions

Retrieves the specified user's printer settings.

If GetUserPrinterOptions cannot retrieve the printer information, for example if the printer is temporarily unavailable, and you specified the PrinterName, the response contains empty or incomplete attributes. If GetUserPrinterOptions cannot retrieve the printer information and you did not specify PrinterName, a SOAP fault is generated.

Request schema

```
<xsd:complexType name="GetUserPrinterOptions">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="UserId" type="xsd:string"/>
         <xsd:element name="UserName" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="PrinterName" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UserId
The ID of the user for whom to retrieve the printer settings. Specify either UserId or UserName.

UserName
The name of the user for whom to retrieve the printer settings. Specify either UserName or UserId.

PrinterName
The name of the printer for which to retrieve the settings.

Response schema

```
<xsd:complexType name="GetUserPrinterOptionsResponse">
   <xsd:sequence>
      <xsd:element name="PrinterOptions"
         type="typens:ArrayOfPrinterOption"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

PrinterOptions
The printer settings.

GetVolumeProperties

Retrieves properties of a specific volume.

Request schema

```
<xsd:complexType name="GetVolumeProperties">
   <xsd:sequence>
      <xsd:element name="ResultDef" type="typens:ArrayOfString">
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ResultDef
The properties to retrieve. By default, VolumeProperties are always returned. In addition, you can specify the following properties:

- ArchiveLibrary
- AutoArchiveSchedule
- ExternalUserPropertyNames
- OnlineBackupSchedule
- TranslatedRoleNames
- PrinterOptions
- VolumeProperties

Response schema

```
<xsd:complexType name="GetVolumePropertiesResponse">
   <xsd:sequence>
      <xsd:element name="VolumeProperties" type="typens:Volume"/>
      <xsd:element name="TranslatedRoleNames"
         type="typens:ExternalTranslatedRoleNames" minOccurs="0"/>
      <xsd:element name="ExternalUserPropertyNames"
         type="typens:ArrayOfString" minOccurs="0"/>
      <xsd:element name="PrinterOptions"
         type="typens:ArrayOfPrinterOptions" minOccurs="0"/>
```

```
      <xsd:element name="AutoArchiveSchedule"
         type="typens:JobSchedule" minOccurs="0"/>
      <xsd:element name="ArchiveLibrary" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ArchiveServiceCmd" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="EventOptions" type="typens:EventOptions"
         minOccurs="0"/>
      <xsd:element name="LicenseOptions"
         type="typens:ArrayOfLicenseOption" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

VolumeProperties
The volume properties.

TranslatedRoleNames
The translated role names.

ExternalUserPropertyNames
The external user properties.

PrinterOptions
The printer options.

AutoArchiveSchedule
The autoarchive schedule.

ArchiveLibrary
The name of the archive application for the Encyclopedia volume.

ArchiveServiceCmd
The archive service command for the Encyclopedia volume.

LicenseOptions
The license options for the Encyclopedia volume.

Login

Authenticates a user to the iHub System.

Request schema

```
<xsd:complexType name="Login">
   <xsd:sequence>
      <xsd:element name="User" type="xsd:string"/>
      <xsd:element name="Password" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="EncryptedPwd" type="xsd:string"
         minOccurs="0"/>
```

```
      <xsd:element name="Credentials" type="xsd:base64Binary"
         minOccurs="0"/>
      <xsd:element name="Domain" type="xsd:string" minOccurs="0"/>
      <xsd:element name="UserSetting" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="ValidateRoles" type="typens:ArrayOfString"
         minOccurs="0"/>
      <xsd:element name="RunAsUser" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

User
The name of the user to log in.

Password
The password of the user to log in. You must specify either Password or Credentials.

EncryptedPwd
The password in encrypted format.

Credentials
Extended credentials data. Used for Report Server Security Extension (RSSE) integration. You must specify Either Credentials or Password.

Domain
The Encyclopedia volume to which to log in. In Release 11, the Login message must specify the Encyclopedia volume name using TargetVolume in the SOAP header and Domain in the SOAP message body.

UserSetting
Specifies whether the response includes detailed user information. If True, the response includes all user attributes. If False, the response does not include detailed user information. The default value is False.

ValidateRole
Checks whether the user has the specified roles.

RunAsUser
Specifies the user name in the run-time environment.

Response schema

```
<xsd:complexType=" LoginResponse">
   <xsd:sequence>
      <xsd:element name="AuthId" type="xsd:string"/>
      <xsd:element name="AdminRights" minOccurs="0">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="Administrator"/>
               <xsd:enumeration value="Operator"/>
```

```
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="User" type="typens:User" minOccurs="0"/>
      <xsd:element name="FeatureOptions"
         type="typens:ArrayOfString" minOccurs="0"/>
      <xsd:element name="ValidRoles" type="typens:ArrayOfString"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

AuthId
The system-generated, encrypted, and authenticated token that the application uses in all subsequent requests.

AdminRights
Returned if the user has administrator rights.

User
All user attributes except the user's password.

FeatureOptions
The features available to the user:

- ReportGeneration
  BIRT option
- PageSecureViewing
  BIRT Page Level Security option

ValidRoles
The user's roles from the ValidateRoles list. Does not return the user's roles that ValidateRoles does not specify. For example, if ValidateRoles specifies Sales, Marketing, and Engineering and the user has Sales and Accounting roles, the response contains only Sales.

MoveFile

Moves files or folders to a new location. To move a single file or folder, specify Name or Id. To move a list of files or folders, specify NameList or IdList. To move files or folders that match the specified conditions, specify Search.

Request schema

```
<xsd:complexType name="MoveFile">
   <xsd:sequence>
      <xsd:element name="Target" type="xsd:string"/>
      <xsd:choice minOccurs="0">
         <xsd:element name="WorkingFolderName" type="xsd:string"/>
         <xsd:element name="WorkingFolderId" type="xsd:string"/>
```

```
      </xsd:choice>
      <xsd:element name="Recursive" minOccurs="0"/>
      <xsd:choice>
         <xsd:element name="Search" type="typens:FileSearch"/>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="NameList" type="typens:ArrayOfString"/>
         <xsd:element name="Id" type="xsd:string"/>
         <xsd:element name="Name" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="ReplaceExisting" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="MaxVersions" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="LatestVersionOnly" type="xsd:Boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Target
The new location for the file or folder. The following rules apply:

- If Target is a file, the operation fails if the source contains a folder.
- If the source is a folder and a folder with the same name exists in the target location, the operation fails.
- If the source is a file and a file with an identical name exists in the target location, the existing file in the target location is versioned or replaced, depending on the setting of the ReplaceExisting tag. If the existing file has any dependencies, the file is not replaced regardless of the ReplaceExisting setting.
- If Target is a folder that does not exist, a folder is created.

WorkingFolderName
The name of the working folder of the file or folder to move. Specify either WorkingFolderName or WorkingFolderId.

WorkingFolderId
The ID of the working folder of the file or folder to move. Specify either WorkingFolderId or WorkingFolderName.

Recursive
Specifies whether to search subfolders. If True, the search includes subfolders. The default value is False.

Search
The search condition that specifies which folders or files to move.

IdList
The list of file or folder IDs to move. Specify either IdList or NameList.

NameList
The list of file or folder names to move. Specify either NameList or IdList.

Id
The ID of the single file or folder to move. Specify either Id or Name.

Name
The name of the single file or folder to move. Specify either Name or Id.

ReplaceExisting
If True, the existing file, if one exists, is replaced. If the existing file has any dependencies, it is not replaced. If False or the existing file has any dependencies, the file is versioned. The default value is True.

MaxVersions
The maximum number of versions to create. MaxVersions applies only if a file is moved. If a folder is moved, MaxVersions is ignored.

LatestVersionOnly
Specifies whether all versions or only the latest version of the file is moved. Used only when a Search tag is specified. If True, only the latest version of the file is moved. The default value is False.

ODBOTunnel

Opens a connection to an OLAP server for ODBO API function.

Request schema

```
<xsd:complexType name="ODBOTunnel">
   <xsd:sequence>
      <xsd:element name="ODBORequest" type="xsd:base64Binary" />
      <xsd:element name="RequestConnectionHandle"
         type="xsd:boolean"
         minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ODBORequest
The ODBORequest

RequestConnectionHandle
A flag indicating whether a connection handle, or session ID for the object, is to be returned.

Response schema

```
<xsd:complexType name="ODBOTunnelResponse">
   <xsd:sequence>
      <xsd:element name="ODBOResponse" type="typens:Attachment"
         minOccurs="0" />
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0" />
```

```
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ODBOResponse
The response from the ODBO API.

ConnectionHandle
The handle to the ODBO connection.

OpenInfoObject

Opens an information object, returning handles to the object in its response.

Request schema

```
<xsd:complexType name="OpenInfoObject">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="ObjectName" type="xsd:string" />
         <xsd:element name="ObjectId" type="xsd:string" />
      </xsd:choice>
      <xsd:element name="Query" type="typens:Query"
         minOccurs="0"/>
      <xsd:element name="DownloadEmbedded" type="xsd:boolean"
         minOccurs="0" />
      <xsd:element name="ReturnDataInBlocks" type="xsd:boolean"
         minOccurs="0" />
      <xsd:element name="FetchSize" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="Format" type="typens:InfoObjectDataFormat"
         minOccurs="0" />
      <xsd:element name="DownloadDoubleAsBinary" type="xsd:boolean"
         minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ObjectName
The name of the information object.

ObjectId
The ID of the information object.

Query
The query name.

ReturnDataInBlocks
A flag indicating whether to return data in blocks as specified for the database. If ReturnDataInBlocks is false, FetchSize is set to 0.

FetchSize
The maximum number of records to retrieve and return in a result set. The default value is 500.

Format
The format of the information object.

DownloadDoubleAsBinary
A flag indicating whether to download double values in binary format.

Response schema

```
<xsd:complexType name="OpenInfoObjectResponse">
   <xsd:sequence>
      <xsd:element name="DataFetchHandle" type="xsd:string" />
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Response elements

DataFetchHandle
The handle to the information object.

ConnectionHandle
The ID of the object. Supports viewing an item already in the iHub System.

Ping

Tests whether a specific component of BIRT iHub is operational and retrieves other diagnostic information about the component.

Request schema

```
<xsd:complexType name="Ping">
   <xsd:sequence>
      <xsd:element name="Destination">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="MDS"/>
               <xsd:enumeration value="EE"/>
               <xsd:enumeration value="FS"/>
               <xsd:enumeration value="VS"/>
               <xsd:enumeration value="OSD"/>
               <xsd:enumeration value="AIS"/>
               <xsd:enumeration value="ACS"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="Action" minOccurs="0">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="Echo"/>
```

```
                <xsd:enumeration value="ReadFile"/>
                <xsd:enumeration value="WriteFile"/>
                <xsd:enumeration value="Connect"/>
             </xsd:restriction>
          </xsd:simpleType>
       </xsd:element>
       <xsd:element name="Mode" minOccurs="0">
          <xsd:simpleType>
             <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Normal"/>
                <xsd:enumeration value="Trace"/>
                <xsd:enumeration value="Concise"/>
             </xsd:restriction>
          </xsd:simpleType>
       </xsd:element>
       <xsd:element name="Server" type="xsd:string" minOccurs="0"/>
       <xsd:element name="ProcessID" type="xsd:string"
          minOccurs="0"/>
       <xsd:element name="FileName" type="xsd:string"
          minOccurs="0"/>
       <xsd:element name="PartitionName" type="xsd:string"
          minOccurs="0"/>
       <xsd:element name="NumBytes" type="xsd:long"
          minOccurs="0"/>
       <xsd:element name="ConnectionProperties"
          type="typens:ArrayOfParameterValue" minOccurs="0"/>
       <xsd:element name="Payload" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
 </xsd:complexType>
```

Request elements

Destination
The component to test. Valid values are:

- MDS
  A Message Distribution service
- EE
  An Encyclopedia engine
- FS
  A Factory service
- VS
  A View service
- OSD
  An open server driver

- AIS
  An Actuate Integration service
- ACS
  An Actuate Caching service

Action
The optional action to take. Valid values are:

- Echo
  Echoes the data specified in Payload.
- ReadFile
  Opens the specified Encyclopedia volume file, reads the file's contents, then closes the file. Applies only if the value of Destination is EE, FS, or VS. Ping returns the timing of the read operation. Specify the file name in FileName.
- WriteFile
  Creates a temporary file on a partition, writes a specified number of bytes to the file, closes the file, then deletes the file. Applies only if the value of Destination is EE or FS. Ping returns the timing information for each step. Specify the partition in PartitionName. Specify the number of bytes to read in NumBytes.
- Connect
  Connects to a data source. Specify the connection parameters in ConnectionProperties.

Mode
The level of detail to return. Valid values are:

- Normal
  Returns the names of components in the test path and the timestamps of the request entering and leaving each component. This is the default mode.
- Trace
  Returns the timestamp of the request entering and leaving major subcomponents of the component being tested.
- Concise
  Returns the elapsed time between a component's receipt of the request and the time the component sends a reply.

Server
Specifies which instance of a Factory service or View service to test. Applies only if the value of Destination is FS or VS. Use Server in conjunction with the ProcessID element. To test all available instances of the Factory or View service,

specify an asterisk (*). If not specified, the BIRT iHub load-balancing mechanism allocates an available instance of the requested service to respond to the request.

ProcessID
Specifies the process ID of the Factory or View service to test. Use in conjunction with the Server element. Applies only if the value of Destination is FS or VS.

FileName
If the value of Action is ReadFile, indicates the Encyclopedia volume file to read. If the value of Destination is OSD, specifies the executable file to prepare for execution.

PartitionName
Specifies the name of the partition on which to create the temporary file. Applies only if the value of Action is WriteFile.

NumBytes
Specifies the number of bytes to read or write. Applies only if the value of Action is ReadFile or WriteFile. If NumBytes is not specified or 0, the default value of 10 KB is used.

ConnectionProperties
An array of property name and value pairs that specify the parameter values for establishing a data source connection. Applies only if the value of Action is Connect. To establish a connection, you must specify a property with a name DBType and a value that specifies the type of database. You must also specify any other properties that the specific database interface requires. Table 8-3 lists the valid property names.

Table 8-3 Valid connection properties

Property name	Applicable database interface	Description
DBType	All	The type of database. Valid values are: ■ DB2, Informix, MSSQL ■ ODBC, Oracle, Sybase, Progress, Progress SQL92
DllPath	DB2, Informix, MSSQL, ODBC, Oracle, Progress, Progress SQL92, Sybase	The name of the DLL providing the client database.
UserName	DB2, Informix, MSSQL, ODBC, Oracle, Progress, Progress SQL92, Sybase	The database user name.

(continues)

Table 8-3 Valid connection properties (continued)

Property name	Applicable database interface	Description
Password	DB2, Informix, MSSQL, ODBC, Oracle, Progress, Progress SQL92, Sybase	The database password.
DataSource	DB2, ODBC	The name of the data source.
ConnectionString	ODBC	Any additional text that ODBC needs to establish the connection.
HostString	Oracle	The Oracle server name for the connection.
DatabaseEnvironment	Informix	The name of the database server, the database, or both database server and database to which to connect.
DatabaseList	Progress	The name of the database.
StartUpParameters	Progress	The Progress Open Interface Broker parameters.
Database	Progress SQL92	The name of the database.
Host	Progress SQL92	The host computer name for a remote database. Not used for a local database. Required if you are connecting to a database running on a database server.
ServiceOrPort	Progress SQL92	The database service name or port number on the database server. Not used for a local database. The port number is an unsigned 16-bit integer in the range 1–65535.

Payload
Specifies the payload data. Applies only if the value of Action is Echo. Payload is binary data attached to the request.

Response schema

```
<xsd:complexType name="PingResponse">
   <xsd:sequence>
      <xsd:element name="Reply" type="xsd:string"/>
      <xsd:element name="Payload" type="xsd:string" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Reply
The Ping reply, in plain text format. The information depends on the value of Mode.

Payload
If a value is specified for Payload in the request, the payload data as a string.

PrintReport

Prints a document. PrintReport requests are always executed in asynchronous mode.

The document prints to the specified printer. If a printer is not specified, the document prints to the user's default printer.

Both PrintReport and SubmitJob support printing. Use PrintReport to print an existing document. Use SubmitJob to schedule execution and printing from a design executable.

Request schema

```
<xsd:complexType name="PrintReport">
   <xsd:sequence>
      <xsd:element name="JobName" type="xsd:string"/>
      <xsd:element name="Priority" type="xsd:int"/>
         <xsd:choice>
            <xsd:element name="InputFileName" type="xsd:string"/>
            <xsd:element name="InputFileId" type="xsd:string"/>
         </xsd:choice>
         <xsd:element name="Schedules" type="typens:JobSchedule"
            minOccurs="0"/>
         <xsd:element name="PrinterOptions"
            type="typens:JobPrinterOptions" minOccurs="0"/>
         <xsd:element name="NotifyUsersByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="NotifyGroupsByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="NotifyChannelsByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="NotifyUsersById"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="NotifyGroupsById"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="NotifyChannelsById"
            type="typens:ArrayOfString"minOccurs="0"/>
         <xsd:element name="SendSuccessNotice"
            type="xsd:boolean" minOccurs="0"/>
         <xsd:element name="SendFailureNotice" type="xsd:boolean"
            minOccurs="0"/>
         <xsd:element name="SendEmailForSuccess"
            type="xsd:boolean" minOccurs="0"/>
         <xsd:element name="SendEmailForFailure"
            type="xsd:boolean" minOccurs="0"/>
```

```
            <xsd:element name="OverrideRecipientPref"
               type="xsd:boolean"
            <xsd:element name="RecordSuccessStatus"
               type="xsd:boolean" minOccurs="0"/>
            <xsd:element name="RecordFailureStatus"
               type="xsd:boolean" minOccurs="0"/>
            <xsd:element name="RetryOptions"
               type="typens:RetryOptions" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
```

Request elements

JobName
The job name.

Priority
The job priority. Limited by the user's Max job priority setting.

InputFileName
The name of the file to print. Specify either InputFileName or InputFileId.

InputFileId
The ID of the file to print. Specify either InputFileId or InputFileName.

Schedules
The schedule for the print job. If not specified, the print request is sent immediately.

PrinterOptions
The job printer settings. The printer settings have the following precedence:

- Job printer settings
- User printer settings
- System printer settings

NotifyUsersByName
The names of users to receive job completion notice. Specify either NotifyUsersByName or NotifyUsersById.

NotifyGroupsByName
The names of groups to receive the job completion notice. Specify either NotifyGroupsByName or NotifyGroupsById.

NotifyChannelsByName
The names of channels to receive the job completion notice. Specify either NotifyChannelsByName or NotifyChannelsById.

NotifyUsersById
The IDs of users to receive the job completion notice. Specify either NotifyUsersById or NotifyUsersByName.

NotifyGroupsById
The IDs of groups to receive the job completion notice. Specify either NotifyGroupsById or NotifyGroupsByName.

NotifyChannelsById
The IDs of channels to receive job completion notice. Specify either NotifyChannelsById or NotifyUsersByName.

SendSuccessNotice
Specifies whether notices are sent if report printing succeeds.

SendFailureNotice
Specifies whether notices are sent if report printing fails.

SendEmailForSuccess
Specifies whether an e-mail is sent when report printing succeeds.

SendEmailForFailure
Specifies whether an e-mail is sent when report printing fails.

OverrideRecipientPref
Specifies whether recipient preferences are overridden.

RecordSuccessStatus
Specifies whether the job status is kept if report printing succeeds.

RecordFailureStatus
Specifies whether the job status is kept if report printing fails.

RetryOptions
Specifies how to retry printing if the previous attempt failed. Used only if Retryable is specified.

Response schema

```
<xsd:complexType name="PrintReportResponse">
   <xsd:sequence>
      <xsd:element name="JobId" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

JobId
The ID of the print job. Returned after the job is created.

SaveSearch

Saves the results of a search.

Request schema

```
<xsd:complexType name="SaveSearch">
   <xsd:sequence>
      <xsd:element name="BasedOnFile"
```

```
            type="typens:ObjectIdentifier" />
         <xsd:element name="SelectList"
            type="typens:ArrayOfComponentIdentifier" />
         <xsd:element name="SearchList" type="typens:ArrayOfComponent"
            minOccurs="0" />
         <xsd:element name="SearchFile" type="typens:NewFile" />
      </xsd:sequence>
   </xsd:complexType>
```

Request elements

BasedOnFile
The file the search takes place in.

SelectList
The list of where to search within the file.

SearchList
The items to search for.

SearchFile
The file the search is saved to.

Response schema

```
<xsd:complexType name="SaveSearchResponse">
   <xsd:sequence>
      <xsd:element name="FileId" type="xsd:string" />
   </xsd:sequence>
</xsd:complexType>
```

Response elements

FileId
The ID of the saved file.

SaveTransientReport

Saves a report to a specified file.

Request schema

```
<xsd:complexType name="SaveTransientReport">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
      <xsd:element name="NewFile" type="typens:NewFile"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The object identifier of the report.

NewFile
The file the report is saved in.

Response schema

```
<xsd:complexType name="SaveTransientReportResponse">
   <xsd:sequence>
      <xsd:element name="FileId" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

FileId
File ID of created report file.

SearchReport

Searches a report for the specified criteria.

Request schema

```
<xsd:complexType name="SearchReport">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
      <xsd:element name="ViewParameter"
         type="typens:ViewParameter"/>
      <xsd:choice minOccurs="0">
         <xsd:element name="SearchReportByNameList"
            type="typens:SearchReportByNameList"/>
         <xsd:element name="SearchReportByIdList"
            type="typens:SearchReportByIdList"/>
         <xsd:element name="SearchReportByIdNameList"
            type="typens:SearchReportByIdNameList"/>
      </xsd:choice>
      <xsd:element name="Range" type="typens:Range"
         minOccurs="0"/>
      <xsd:element name="DownloadEmbedded" type="xsd:boolean"
         default="false" minOccurs="0"/>
      <xsd:element name="OutputProperties"
         type="typens:SearchResultProperties" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The ID of the object to search.

ViewParameter
The viewing parameters. SearchReport uses a different set of formats than other operations that use the ViewParameter data type. Valid formats are:

- CSV
- EXCEL
- TSV

- UNCSV
- UNTSV
- XMLDisplay

Optionally, you can specify the UserAgent. UserAgent specifies which browser to use for report viewing.

Range
The range containing the first and last record number. The range starts at 0.

DownloadEmbedded
Specifies whether to send the attachment embedded in the body of the SOAP message or in a separate MIME boundary. If True, the response contains the attachment as embedded data. If False, the attachment is in a separate MIME boundary. The default value is False.

OutputProperties
Applies only if Format is CSV. The properties to include in the search result are:

- EnableColumnHeaders
  If True, column headers are included in the search result. The default value is True.
- UseQuoteDelimiter
  If True, each data item in the search result is enclosed in double quotes (" "). The default value is True.

Response schema

```
<xsd:complexType name="SearchReportResponse">
   <xsd:sequence>
      <xsd:element name="SearchRef" type="typens:Attachment"/>
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ReportType" minOccurs="0">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="ActuateBasic"/>
               <xsd:enumeration value="ActuateBasicInfoObj"/>
               <xsd:enumeration value="InfomationObject"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

SearchRef
The details of the attachment such as contentId, contentType, contentLength, contentEncoding, locale, and contentData.

ConnectionHandle
The ID of the report. Supports viewing a report when the report is already in iHub System. Specified in the SOAP header.

ReportType
The report type. Valid report types are:

- ActuateBasic
- ActuateBasicInfoObj
- InformationObject

SelectChannels

Searches channels for specified information.

To search a single channel, specify Name or Id. To search a list of channels, specify NameList or IdList. To search channels matching the specified conditions, specify Search.

Request schema

```
<xsd:complexType name="SelectChannels">
   <xsd:sequence>
      <xsd:element name="ResultDef" type="typens:ArrayOfString"/>
         <xsd:choice>
            <xsd:element name="Search"
               type="typens:ChannelSearch"/>
            <xsd:element name="NameList"
               type="typens:ArrayOfString"/>
            <xsd:element name="IdList" type="typens:ArrayOfString"/>
            <xsd:element name="Name" type="xsd:string"/>
            <xsd:element name="Id" type="xsd:string"/>
         </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ResultDef
The properties to retrieve.

Search
The search conditions. If conditions apply to multiple fields, use ConditionArray.

NameList
The list of channel names to search. Specify either NameList or IdList.

IdList
The list of channel IDs to search. Specify either IdList or NameList.

Name
The name of a single channel to search. Specify either Name or Id.

Id
The ID of the single channel to search. Specify either Id or Name.

Response schema

```
<xsd:complexType name="SelectChannelsResponse">
   <xsd:sequence>
      <xsd:element name="Channels" type="typens:ArrayOfChannel"/>
      <xsd:element name="FetchHandle" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="TotalCount" type="xsd:long"\
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Channels
The selected channels.

FetchHandle
Indicates that the number of items in the result set exceeds the FetchSize limit.

TotalCount
The number of entries in the search result set.

SelectFiles

Retrieves information about a specified file.

To retrieve a single file or folder, specify Name or Id. To retrieve a list of files or folders, specify NameList or IdList. To search all file or folders that match specific condition, specify Search.

Request schema

```
<xsd:complexType name="SelectFiles">
   <xsd:sequence>
      <xsd:choice minOccurs="0">
         <xsd:element name="WorkingFolderId" type="xsd:string"/>
         <xsd:element name="WorkingFolderName"
               type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="Recursive" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="LatestVersionOnly" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="ResultDef" type="typens:ArrayOfString"/>
      <xsd:choice>
         <xsd:element name="Search" type="typens:FileSearch"/>
         <xsd:element name="NameList"
            type="typens:ArrayOfString"/>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="Name" type="xsd:string"/>
```

```
            <xsd:element name="Id" type="xsd:string"/>
         </xsd:choice>
         <xsd:element name="Content" minOccurs="0">
            <xsd:simpleType>
               <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="Embed"/>
                  <xsd:enumeration value="Attach"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
      </xsd:sequence>
   </xsd:complexType>
```

Request elements

WorkingFolderId
The ID of the working folder in which to search for the file. Specify either WorkingFolderId or WorkingFolderName.

WorkingFolderName
The absolute path and the name of the working folder in which to search for the file. Specify either WorkingFolderName or WorkingFolderId.

Recursive
Specifies whether to search subfolders. If True, the search includes subfolders. The default value is False.

LatestVersionOnly
Specifies whether to search all versions of the file. If True, the search includes only the latest version. The default value is False.

ResultDef
The properties to retrieve.

Search
The search condition. If conditions apply to multiple fields, use ConditionArray.

NameList
The name of the file or folder list to retrieve. Specify either IdList or NameList.

IdList
The ID of the file or folder list to retrieve. Specify either IdList or NameList.

Name
The name of a single file or folder to retrieve. Specify either Name or Id.

Id
The ID of a file or folder to retrieve. Specify either Id or Name.

Content
Specifies whether the file is embedded in or attached to the response. Valid values are:

- Embed
  The file is embedded.
- Attach
  The file is attached.

Response schema

```
<xsd:element name="SelectFilesResponse">
   <xsd:sequence>
      <xsd:element name="ItemList" type="typens:ArrayOfFile"/>
      <xsd:element name="ContentItemList"
         type="typens:ArrayOfFileContent"/>
      <xsd:element name="FetchHandle" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="TotalCount" type="xsd:long"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ItemList
The list of attached items that match the search criteria.

ContentItemList
The list of embedded items that match the search criteria.

FetchHandle
Indicates that the number of items in the result set exceeds the FetchSize limit.

TotalCount
The number of entries in the search result set.

SelectFileTypes

Searches file types for specified information.

To search a single file type, specify Name. To search a list of file types, specify NameList. To search file types matching the specified conditions, specify Search. File names are case sensitive, and file type extensions are stored in uppercase. Specify uppercase for all file type extensions, for example, use RPTDESIGN instead of rptdesign.

Request schema

```
<xsd:complexType name="SelectFileTypes">
   <xsd:sequence>
      <xsd:element name="ResultDef" type="typens:ArrayOfString"/>
         <xsd:choice minOccurs="0">
```

```
            <xsd:element name="NameList"
               type="typens:ArrayOfString"/>
            <xsd:element name="Name" type="xsd:string"/>
         </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ResultDef
The properties to retrieve.

Search
The search conditions. If conditions apply to multiple fields, use ConditionArray.

NameList
The list of file types to search.

Name
The name of a single file type to search.

Response schema

```
<xsd:complexType name="SelectFileTypesResponse">
   <xsd:sequence>
      <xsd:element name="FileTypes" type="typens:ArrayOfFileType"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

FileTypes
The specified file types.

SelectGroups

Searches groups for specified information.

To search a single group, specify Name or Id. To search a list of groups, specify NameList or IdList. To search groups matching the specified conditions, specify Search.

Request schema

```
<xsd:complexType name="SelectGroups">
   <xsd:sequence>
      <xsd:element name="ResultDef" type="typens:ArrayOfString"/>
         <xsd:choice>
            <xsd:element name="Search" type="typens:GroupSearch"/>
            <xsd:element name="IdList" type="typens:ArrayOfString"/>
            <xsd:element name="NameList"
               type="typens:ArrayOfString"/>
            <xsd:element name="Id" type="xsd:string"/>
            <xsd:element name="Name" type="xsd:string"/>
         </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ResultDef
The properties to retrieve.

Search
The search conditions. If conditions apply to multiple fields, use ConditionArray. When using RSSE external registration, SelectGroups allows only one search condition.

IdList
The list of group IDs to search. Specify either IdList or NameList.

NameList
The list of group names to search. Specify either NameList or IdList.

Id
The ID of the single group to search. Specify either Id or Name.

Name
The name of the single group to search. Specify either Name or Id.

Response schema

```
<xsd:complexType name="SelectGroupsResponse">
   <xsd:sequence>
      <xsd:element name="Groups" type="typens:ArrayOfGroup"/>
      <xsd:element name="FetchHandle" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="TotalCount" type="xsd:long"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Groups
The selected groups.

FetchHandle
Indicates that the number of items in the result set exceeds the FetchSize limit.

TotalCount
The number of entries in the search result set.

SelectJavaReportPage

Returns a report page formatted in the specified display format indicated by the Page or Component element.

Request schema

```
<xsd:complexType name="SelectJavaReportPage">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
      <xsd:choice>
```

```
            <xsd:element name="Page" type="typens:PageIdentifier"
               minOccurs="0"/>
            <xsd:element name="Component"
               type="typens:ArrayOfNameValuePair"minOccurs="0"/>
         </xsd:choice>
         <xsd:element name="ViewParameterValues"
               type="typens:ArrayOfParameterValue" minOccurs="0"/>
         <xsd:element name="OutputFormat" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="ViewProperties"
            type="typens:ArrayOfNameValuePair"minOccurs="0"/>
         <xsd:element name="DownloadEmbedded" type="xsd:boolean"
            default="false" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
```

Request elements

Object
The ID of the object from which to select the report page.

Page
The identifier of the page to retrieve.

Component
The name, display name, or ID, and the value of the component from which to retrieve the data.

ViewParameterValues
Include view parameters if defined in the document. This feature is not currently supported for a BIRT report.

OutputFormat
For a BIRT report, the output format can be HTML, rptdocument, or PDF. The default is rptdocument.

ViewProperties
Specifies the layout and contents of a report such as the name of a bookmark name, table of contents, or an object ID. ViewProperties is available to the BIRT render task as the java.util.Map object in the Engine AppContext under the key ServerViewProperties.

DowloadEmbedded
Specifies whether to send the attachment embedded in the body of the SOAP message or in a separate MIME boundary. If True, the response contains the attachment as embedded data. If False, the attachment is in a separate MIME boundary. The default value is False.

Response schema

```
<xsd:complexType name="SelectJavaReportPageResponse">
   <xsd:sequence>
      <xsd:element name="PageRef" type="typens:Attachment"/>
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="OutputProperties"
         type="typens:ArrayOfNameValuePair" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

PageRef
Contains the following properties of the attachment:

- MimeType
- ContentEncoding
- ContentLength
- Locale

ConnectionHandle
The ID of the report. Supports viewing a report when the report is already in iHub System. Specified in the SOAP header.

OutputProperties
Applies only if Format is CSV. The properties to include in the search result are:

- EnableColumnHeaders
  If True, column headers are included in the search result. The default value is True.
- UseQuoteDelimiter
  If True, each data item in the search result is enclosed in double quotes (" "). The default value is True.

SelectJobs

Selects all jobs matching the specified conditions. SelectJobs returns states and information for job instances.

In Release 11 or iHub, use SelectJobSchedules to obtain job schedule information. In Release 10 Service Pack 1 and earlier, use SelectJobs with JobCondition, specifying Scheduled, Expired, or Cancelled, to obtain designated job schedule information.

Release 11 Service Pack 4 and later support backward compatibility for all pre-Release 11 applications that send a SelectJobs request. However, all Release 11

and iHub applications must implement the new SelectJobSchedules API to request schedule information.

Request schema

```
<xsd:complexType name="SelectJobs">
   <xsd:sequence>
      <xsd:element name="ResultDef" type="typens:ArrayOfString"/>
      <xsd:choice>
         <xsd:element name="Search" type="typens:JobSearch"/>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="Id" type="xsd:string"/>
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ResultDef
The properties to retrieve.

Search
The search conditions. If conditions apply to multiple fields, use ConditionArray.

IdList
The list of job IDs to search.

Id
The ID of a single job to search.

Response schema

```
<xsd:complexType name="SelectJobsResponse">
   <xsd:sequence>
      <xsd:element name="Jobs" type="typens:ArrayOfJobProperties"/>
      <xsd:element name="FetchHandle" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="TotalCount" type="xsd:long"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Jobs
The selected jobs.

FetchHandle
Indicates that the number of items in the result set exceeds the FetchSize limit.

TotalCount
The number of entries in the search result set.

SelectJobNotices

Retrieves job notices matching the specified criteria.

Request schema

```
<xsd:complexType="SelectJobNotices">
   <xsd:sequence>
      <xsd:element name="ResultDef" type="typens:ArrayOfString">
      </xsd:element>
      <xsd:element name="Search" type="typens:JobNoticeSearch"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ResultDef
The properties to retrieve. You can specify the following properties:

- JobId
- JobName
- ActualHeadline
- CompletionTime
- ActualOutputFileId
- ActualOutputFileName
- VersionName
- OutputFileSize

Search
The search conditions. If conditions apply to multiple fields, use ConditionArray.

Response schema

```
<xsd:complexType name="SelectJobNoticesResponse">
   <xsd:sequence>
      <xsd:element name="JobNotices"
         type="typens:ArrayOfJobNotice"/>
      <xsd:element name="FetchHandle" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="TotalCount" type="xsd:long"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

JobNotices
The selected job notices.

FetchHandle
Indicates that the number of items in the result set exceeds the FetchSize limit.

TotalCount
The number of entries in the search result set.

SelectJobSchedules

Selects all scheduled jobs matching the specified criteria. SelectJobSchedules also retrieves information for expired and cancelled jobs.

In Release 11 or iHub, use SelectJobSchedules to obtain job schedule information. In Release 10 Service Pack 1 and earlier, use SelectJobs with JobCondition, specifying Scheduled, Expired, or Cancelled, to obtain designated job schedule information.

Release 11 Service Pack 4 and iHub support backward compatibility for all pre-Release 11 applications that send a SelectJobs request. However, all Release 11 and iHub applications must implement the new SelectJobSchedules API to request schedule information.

Request schema

```
<xsd:complexType name="SelectJobSchedules">
   <xsd:sequence>
      <xsd:element name="ResultDef" type="typens:ArrayOfString">
      </xsd:element>
      <xsd:choice>
         <xsd:element name="Search"
            type="typens:ScheduledJobSearch"/>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="Id" type="xsd:string"/>
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ResultDef
The properties to retrieve.

Search
The search conditions. If conditions apply to multiple fields, use ConditionArray.

IdList
The list of job IDs to search.

Id
The ID of a single job to search.

Response schema

```
<xsd:complexType name="SelectJobSchedulesResponse">
   <xsd:sequence>
      <xsd:element name="Jobs" type="typens:ArrayOfJobProperties"/>
      <xsd:element name="FetchHandle" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="TotalCount" type="xsd:long"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Jobs
The selected jobs.

FetchHandle
Indicates that the number of items in the result set exceeds the FetchSize limit.

TotalCount
The number of entries in the search result set.

SelectPage

Retrieves a page or a set of pages. You can specify either a page number, a page range, a component, or other search criteria. Specify FileType in the SOAP header for all execute, submit, and view IDAPI requests.

The response to SelectPage contains the following data:

- The SOAP response.
- The attachment containing the data.
- If the request input format is Reportlet, an XML response as an attachment. This response contains the post process data.

Request schema

```
<xsd:complexType name="SelectPage">
   <xsd:sequence>
      <xsd:element name="Object" type="typens:ObjectIdentifier"/>
      <xsd:element name="ViewParameter"
         type="typens:ViewParameter"/>
      <xsd:choice minOccurs="0">
         <xsd:element name="Page" type="typens:PageIdentifier"/>
         <xsd:element name="Component" type="typens:Component"/>
         <xsd:element name="SearchCriteria" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="MaxHeight"
         type="xsd:long" minOccurs="0"/>
      <xsd:element name="CustomInputPara" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="DownloadEmbedded" type="xsd:boolean"
         default="false" minOccurs="0"/>
      <xsd:element name="SplitOversizePages" type="xsd:boolean"
         minOccurs="0"
      <xsd:element name="PageWidth" type="xsd:int"
         minOccurs="0"/>
      <xsd:element name="PageHeight" type="xsd:int"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Object
The object from which to retrieve the page. Specify either the object ID or the object name and version number.

ViewParameter
The viewing parameters.

Page
The page or range of pages to retrieve.

Component
The name, display name, or ID, and the value of the component from which to retrieve the page. The following formats do not support specifying the component ID:

- ExcelData
- ExcelDisplay
- PDF
- RTF

SearchCriteria
The search criteria.

MaxHeight
The maximum height, in points, according to the web page layout design. Required for a Reportlet.

CustomInputPara
The input parameters to send.

DownloadEmbedded
Specifies whether to send the attachment embedded in the body of the SOAP message or in a separate MIME boundary. If True, the response contains the attachment as embedded data. If False, the attachment is in a separate MIME boundary. The default value is False.

SplitOversizePages
Specifies whether to split a page to print to an output format that is smaller than the page. If True, the page is split. The default value is True.

PageWidth
The page width page to use for printing the page.

PageHeight
The page height page to use for printing the page.

Response schema

```
<xsd:complexType name="SelectPageResponse">
   <xsd:sequence>
      <xsd:element name="PageRef" type="typens:Attachment"/>
      <xsd:element name="PostResponseRef" type="typens:Attachment"
         minOccurs="0"/>
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

PageRef
Contains the following properties of the attachment:

- MimeType
- ContentEncoding
- ContentLength
- Locale

PostResponseRef
Contains the Reportlet parameters. Used PostResponseRef only when the request input format is Reportlet.

ConnectionHandle
The ID of the report. Supports viewing a report when the report is already in iHub System. Specified in the SOAP header.

SelectRoles

Searches roles for specified information.

To search a single role, specify Name or Id. To search a list of roles, specify NameList or IdList. To search roles matching the specified conditions, specify Search.

Request schema

```
<xsd:complexType="SelectRoles">
   <xsd:sequence>
      <xsd:element name="ResultDef" type="typens:ArrayOfString"/>
      <xsd:choice>
         <xsd:element name="Search" type="typens:RoleSearch"/>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="NameList" type="typens:ArrayOfString"/>
         <xsd:element name="Id" type="xsd:string"/>
         <xsd:element name="Name" type="xsd:string"/>
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ResultDef
The properties to retrieve.

Search
The search conditions. If conditions apply to multiple fields, use ConditionArray. When using RSSE external registration, SelectRoles allows only one search condition within the Search element.

In the following example, <Condition> and <AssignedToUserName> are search conditions:

```
<Search>
   <Condition>
      <Field>Name</Field>
      <Match>roleA</Match>
   </Condition>
   <ConditionArray xsi:nil="true"/>
   <ParentRoleName xsi:nil="true"/>
   <ChildRoleName xsi:nil="true"/>
   <WithRightsToChannelName xsi:nil="true"/>
   <AssignedToUserName>Administrator</AssignedToUserName>
   <ParentRoleId xsi:nil="true"/>
   <ChildRoleId xsi:nil="true"/>
   <WithRightsToChannelId xsi:nil="true"/>
   <AssignedToUserId xsi:nil="true"/>
</Search>
```

When using RSSE external registration, the previous search pattern is invalid, generating the following response:

```
<Description>
   <Message>The search pattern is too long or is incorrect.
      </Message>
   <Parameter1>More than one search condition is invalid under
      External Registration</Parameter1>
</Description>
```

IdList
The list of role IDs to search. Specify either IdList or NameList.

NameList
The list of role names to search. Specify either NameList or IdList.

Id
The ID of the single role to search. Specify either Id or Name.

Name
The name of the single role to search. Specify either Name or Id

Response schema

```
<xsd:complexType name="SelectRolesResponse">
   <xsd:sequence>
      <xsd:element name="Roles" type="typens:ArrayOfRole"/>
      <xsd:element name="FetchHandle" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="TotalCount" type="xsd:long"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Roles
The selected roles.

FetchHandle
Indicates that the number of items in the result set exceeds the FetchSize limit.

TotalCount
The number of entries in the search result set.

SelectUsers

Searches users for specified information.

To search a single user, specify Name or Id. To search a list of users, specify NameList or IdList. To search users matching the specified conditions, specify Search.

Request schema

```
<xsd:complexType name="SelectUsers">
   <xsd:sequence>
      <xsd:element name="ResultDef" type="typens:ArrayOfString"/>
      <xsd:choice minOccurs="0">
         <xsd:element name="Search" type="typens:UserSearch"/>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="NameList" type="typens:ArrayOfString"/>
         <xsd:element name="Id" type="xsd:string"/>
         <xsd:element name="Name" type="xsd:string"/>
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ResultDef
The properties to retrieve.

Search
The search conditions. If conditions apply to multiple fields, use ConditionArray. When using RSSE external registration, SelectUsers allows only one search condition.

IdList
The list of user IDs to search. Specify either IdList or NameList.

NameList
The list of user names to search. Specify either NameList or IdList.

Id
The ID of the single user to search. Specify either Id or Name.

Name
The name of the single user to search. Specify either Name or Id.

Response schema

```
<xsd:complexType name="SelectUsersResponse">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="Users" type="typens:ArrayOfUser"/>
         <xsd:element name="FetchHandle" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="TotalCount" type="xsd:long"
            minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
```

Response elements

Users
The selected users.

FetchHandle
Indicates that the number of items in the result set exceeds the FetchSize limit.

TotalCount
The number of entries in the search result set. Not used when querying an Encyclopedia volume that uses Open Security. In this case, TotalCount returns Null. To retrieve the number of entries from an Encyclopedia volume that uses Open Security, use an array length. For example, the following code returns Null:

```
com.actuate.schemas.SelectUsersResponse userSrchResponse =
   proxy.selectUsers( userSel );
com.actuate.schemas.ArrayOfUser userArr =
   userSrchResponse.getUsers();
com.actuate.schemas.User []user = userArr.getUser();
userSrchResponse.getTotalCount();
int noOfUsers1 = userSrchResponse.getTotalCount().intValue();
The following code returns the correct value:
com.actuate.schemas.SelectUsersResponse userSrchResponse =
   proxy.selectUsers( userSel );
com.actuate.schemas.ArrayOfUser userArr =
   userSrchResponse.getUsers();
com.actuate.schemas.User []user = userArr.getUser();
userSrchResponse.getTotalCount();
int noOfUsers2 = user.length;
```

SetConnectionProperties

Sets the connection properties for a file based on user or role.

Request Schema

```
<xsd:complexType name="SetConnectionProperties">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="FileName" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="FileId" type="xsd:string"
            minOccurs="0"/>
      </xsd:choice>
      <xsd:choice>
         <xsd:element name="UserName" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="UserId" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="RoleName" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="RoleId" type="xsd:string"
            minOccurs="0"/>
      </xsd:choice>
      <xsd:element name="ConnectionProperties"
         type="typens:ArrayOfPropertyValue" />
   </xsd:sequence>
</xsd:complexType>
```

Request elements

FileName
The name of the file.

FileId
The file ID.

UserName
User name for which to set property values.

UserId
User ID for which to set property values.

RoleName
The role name for which to set property values.

RoleId
The role ID for which to set property values.

ConnectionProperties
An array of name-value pairs containing the connection properties being set.

Response Schema

```
<xsd:complexType name="SetConnectionPropertiesResponse" />
```

SetServerResourceGroupConfiguration

Sets or updates properties of all resource groups on a BIRT iHub. SetServerResourceGroupConfiguration is available only to an Encyclopedia volume administrator or a user with an Administrator role.

Request schema

```
<xsd:complexType="SetServerResourceGroupConfiguration">
   <xsd:sequence>
      <xsd:element name="ServerName" type="xsd:string"/>
      <xsd:element name="ServerResourceGroupSettingList"
         type="typens:ArrayOfServerResourceGroupSetting"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ServerName
The name of the BIRT iHub.

ServerResourceGroupSettingList
Contains one or more of the following properties to set or update:

- Activate
- MaxFactory
- FileTypes
  Cannot be set or updated for a default resource group

Response schema

```
<xsd:complexType name=
      "SetServerResourceGroupConfigurationResponse"/>
   <xsd:sequence/>
</xsd:complexType>
```

SubmitJob

Generates and prints a report or information object in asynchronous mode. BIRT iHub sends a response after completing the request.

After generating a document in asynchronous mode, you can convert the document to one of the following formats:

- Advanced function printing (AFP)
- Comma-separated values (CSV)

- Excel XLS
- Excel XLSX
- PDF
- PostScript
- PowerPoint (PPT)
- PowerPoint (PPTX)
- PSV
- Tab-separated values (TSV)
- Word (DOC)
- Word (DOCX)

When converting a document, the properties of the converted file are the same as the original document properties.

BIRT iHub only supports converting document output for asynchronous generation. Conversion is not supported for the following types of document output:

- Synchronous generation
- Report bursting
- Page-level security

Both SubmitJob and PrintReport support printing. Use SubmitJob to schedule execution and printing from design executables. Use PrintReport to print an existing document.

Request schema

```
<xsd:complexType name="SubmitJob">
   <xsd:sequence>
      <xsd:element name="JobName" type="xsd:string"/>
      <xsd:element name="Headline" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Priority" type="xsd:int"/>
      <xsd:element name="ResourceGroup" type="xsd:string"
         minOccurs="0"/>
      <xsd:choice>
         <xsd:element name="InputFileName" type="xsd:string"/>
         <xsd:element name="InputFileId" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="RunLatestVersion" type="xsd:boolean"
         default="true" minOccurs="0"/>
      <xsd:element name="RequestedOutputFile"
         type="typens:NewFile" minOccurs="0"/>
      <xsd:element name="Operation">
```

```
      <xsd:simpleType>
         <xsd:restriction base="xsd:string">
            <xsd:enumeration value="RunReport"/>
            <xsd:enumeration value="RunAndPrintReport"/>
            <xsd:enumeration value="ConvertReport"/>
            <xsd:enumeration value="PrintReport"/>
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:element>
   <xsd:choice minOccurs="0">
      <xsd:element name="ParameterValues"
         type="typens:ArrayOfParameterValue"/>
      <xsd:element name="ParameterValueFileName"
         type="xsd:string"/>
      <xsd:element name="ParameterValueFileId"
         type="xsd:string"/>
      <xsd:element name="Query" type="typens:Query"/>
      <xsd:element name="QueryFileName" type="xsd:string"/>
      <xsd:element name="QueryFileId" type="xsd:string"/>
   </xsd:choice>
   <xsd:element name="Schedules" type="typens:JobSchedule"
      minOccurs="0"/>
   <xsd:element name="PrinterOptions"
      type="typens:JobPrinterOptions" minOccurs="0"/>
   <xsd:element name="NotifyUsersByName"
      type="typens:ArrayOfString" minOccurs="0"/>
   <xsd:element name="NotifyGroupsByName"
      type="typens:ArrayOfString" minOccurs="0"/>
   <xsd:element name="NotifyChannelsByName"
      type="typens:ArrayOfString" minOccurs="0"/>
   <xsd:element name="NotifyUsersById"
      type="typens:ArrayOfString" minOccurs="0"/>
   <xsd:element name="NotifyGroupsById"
      type="typens:ArrayOfString"MinOccurs="0"/>
   <xsd:element name="NotifyChannelsById"
      type="typens:ArrayOfString" minOccurs="0"/>
   <xsd:element name="SendSuccessNotice" type="xsd:boolean"
      minOccurs="0"/>
   <xsd:element name="SendFailureNotice" type="xsd:boolean"
      minOccurs="0"/>
   <xsd:element name="SendEmailForSuccess" type="xsd:boolean"
      minOccurs="0"/>
   <xsd:element name="SendEmailForFailure" type="xsd:boolean"
      minOccurs="0"/>
   <xsd:element name="AttachReportInEmail" type="xsd:boolean"
      minOccurs="0"/>
   <xsd:element name="OverrideRecipientPref" type="xsd:boolean"
```

```
            minOccurs="0"/>
         <xsd:element name="EmailFormat" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="RecordSuccessStatus" type="xsd:boolean"
            minOccurs="0"/>
         <xsd:element name="RecordFailureStatus" type="xsd:boolean"
            minOccurs="0"/>
         <xsd:element name="IsBundled" type="xsd:boolean"
            minOccurs="0"/>
         <xsd:element name="RetryOptions" type="typens:RetryOptions"
            minOccurs="0"/>
         <xsd:element name="OpenServerOptions"
            type="typens:OpenServerOptions" minOccurs="0"/>
         <xsd:element name="KeepOutputFile"
            type="xsd:boolean" minOccurs="0"/>
         <xsd:element name="ConversionOptions"
            type="typens:ConversionOptions" minOccurs="0"/>
         <xsd:element name="WaitForEvent" type="typens:Event"
            minOccurs="0"/>
         <xsd:element name="DataACL" type="typens:ArrayOfString"
            minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
```

Request elements

JobName
The name of the job.

Headline
The job headline.

Priority
The job priority. Job priority is limited by the user's Max job priority setting. Valid values are 0–1,000, where 1,000 is the highest priority.

ResourceGroup
The resource group to which to assign the job. Available only to an Encyclopedia volume administrator or a user with the Administrator role.

InputFileName
The full path, name, and version number of the file to use as input. If RunLatestVersion is specified, the version number is ignored.

InputFileId
The ID of the file to use as input.

RunLatestVersion

Specifies whether to run or print the most recent version of the executable file. If True, the latest version of the file is used. If True, the version number in InputFileName is ignored.The default value is True.

RequestedOutputFile
The file name and extension for the output.

Operation
Specifies the type of task to perform, RunReport, RunAndPrintReport, ConvertReport, or PrintReport.

ParameterValues
A list of parameter values to use. Specify either ParameterValues, ParameterValueFileId, ParameterValueFileName.

ParameterValueFileName
The name of the parameter value file to use. Specify either ParameterValues, ParameterValueFileId, or ParameterValueFileName.

ParameterValueFileId
The ID of the report object values file to use. Specify either ParameterValues, ParameterValueFileId, or ParameterValueFileName.

Schedules
Specifies the schedule on which to run the report. If not specified, the job runs immediately.

PrinterOptions
If the job is to be printed, specifies the job printer settings. If the job is not to be printed, PrinterOptions is ignored. The printer settings have the following precedence:

- Job printer settings
- User printer settings
- System printer settings

NotifyUsersByName
The names of users to receive the job completion notice. Specify either NotifyUsersByName or NotifyUsersById.

NotifyGroupsByName
The names of groups to receive the job completion notice. Specify either NotifyGroupsByName or NotifyGroupsById.

NotifyChannelsByName
The names of channels to receive the job completion notice. Specify either NotifyChannelsByName or NotifyChannelsById.

NotifyUsersById
The IDs of users to receive the job completion notice. Specify either NotifyUsersById or NotifyUsersByName.

NotifyGroupsById
The IDs of groups to receive the job completion notice. Specify either NotifyGroupsById or NotifyGroupsByName.

NotifyChannelsById
The IDs of channels to receive the job completion notice. Specify either NotifyChannelsById or NotifyUsersByName.

SendSuccessNotice
Specifies whether success notices are sent if the job succeeds. Used only if OverrideRecipientPref is True. If SendSuccessNotice is True, notices are sent.

SendFailureNotice
Specifies whether failure notices are sent if the job fails. Used only if OverrideRecipientPref is True. If SendFailureNotice is True, failure notices are sent to specified users and groups if the job fails.

SendEmailForSuccess
Specifies whether e-mail notifications are sent if the job succeeds. Used only if OverrideRecipientPref is True. If True, e-mail notifications are sent to specified users and groups if the job succeeds. The default value is False.

SendEmailForFailure
Specifies whether e-mail notifications are sent to specified users and groups if the job fails. Used only if OverrideRecipientPref is True. If SendEmailForFailure is True, e-mail notifications are sent. The default value is False.

AttachReportInEmail
Specifies whether to attach a report to an e-mail completion notice. Used only if OverrideRecipientPref is True. If AttachReportInEmail is True, the output file is attached to the e-mail notification if the job succeeds. If False, only a link to the output file is sent. Specify the format for the attachment in the EmailFormat parameter. The default value is False.

OverrideRecipientPref
Specifies whether e-mail notifications and output attachments are sent according to job settings or user settings. If True, e-mail notifications and output attachments are sent according to job settings. If False, e-mail notifications and output attachments are sent according to user settings. The default value is False. If False, the following elements are ignored:

- AttachReportInEmail
- SendEmailForSuccess
- SendEmailForFailure
- SendSuccessNotice
- SendFailureNotice

EmailFormat
Specifies the output format of the report attached to the e-mail notification. The following formats are supported:

- ExcelDisplay
- PDF
- rptdesign
- rptdocument

RecordSuccessStatus
Specifies whether to keep the job status for successful jobs. If True, the job status is kept if job execution succeeds.

RecordFailureStatus
Specifies whether to keep the job status for failed jobs. If True, the job status is kept if job execution fails.

IsBundled
Specifies whether a report is bundled. If True, the report is bundled. If False, the report is not bundled. The default value is False.

RetryOptions
Specifies how to retry the job if the previous attempt failed. Used only if Retryable is specified.

OpenServerOptions
Contains the following open server options:

- KeepWorkingSpace
  Specifies whether the workspace directory is removed after the job completes.
- DriverTimeout
  The time for the driver to return from executing a job.
- PollingInterval
  The time interval for the open server to get status messages. The minimum value is 10 seconds.

KeepOutputFile
Specifies whether the generated output file remains in the Encyclopedia volume if the generation request succeeds but the printing request fails. Used if Operation is RunAndPrintReport. If True, the output file remains in the Encyclopedia volume if the printing request fails. If False, the output file is deleted if the printing request fails. The default value is False.

ConversionOptions
Specifies the options for converting BIRT document (.rptdocument) output to another format.

WaitForEvent
An event that must be completed before the response is processed.

DataACL
Specifies the access control list (ACL) restricting data privileges.

Response schema

```
<xsd:complexType name="SubmitJobResponse">
   <xsd:sequence>
      <xsd:element name="JobId" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

JobId
The job ID. Use the job ID to refer to the job in subsequent requests during the current session.

SystemLogin

Logs the user in as the BIRT iHub administrator.

Request schema

```
<xsd:complexType name="SystemLogin">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="SystemPassword" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="SystemPasswordEncryptLevel"
            type="xsd:long" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

Request elements

SystemPassword
The password.

SystemPasswordEncryptLevel
The encryption level of the SystemPassword. Valid values are:

- 0 - No encryption
- 1 - Two way encryption
- 2 - Hash encryption

The default is hash encryption.

Response schema

```
<xsd:complexType name="SystemLoginResponse">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="AuthId" type="xsd:string"/>
```

```
      </xsd:sequence>
   </xsd:complexType>
```

Response elements

AuthId
The system-generated, encrypted authenticated string all subsequent requests use.

Transaction

A packaging mechanism for Administrate operations. If a failure occurs anywhere in a transaction, all operations in the transaction fail.

Request schema

```
<xsd:complexType name="Transaction">
   <xsd:sequence>
      <xsd:element name="TransactionOperation"
         type="typens:TransactionOperation" maxOccurs="unbounded"
         minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Request element

TransactionOperation
The transaction operation.

TransactionOperation

Controls the ability to create, delete, update, copy, and move items within an Encyclopedia volume. A TransactionOperation request represents a single unit of work within a Transaction. Only an Encyclopedia volume administrator or a user in the Administrator role uses these operations.

Request schema

```
<xsd:complexType name="TransactionOperation">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="CreateUser" type="typens:CreateUser"
            minOccurs="0" />
         <xsd:element name="DeleteUser" type="typens:DeleteUser"
            minOccurs="0" />
         <xsd:element name="UndeleteUser" ype="typens:UndeleteUser"
            minOccurs="0"/>
         <xsd:element name="UpdateUser" type="typens:UpdateUser"
            minOccurs="0" />
         <xsd:element name="CreateGroup" type="typens:CreateGroup"
            minOccurs="0" />
         <xsd:element name="DeleteGroup" type="typens:DeleteGroup"
            minOccurs="0" />
         <xsd:element name="UpdateGroup" type="typens:UpdateGroup"
```

```
            minOccurs="0" />
          <xsd:element name="CreateChannel"
            type="typens:CreateChannel"
            minOccurs="0" />
          <xsd:element name="DeleteChannel"
            type="typens:DeleteChannel"
            minOccurs="0" />
          <xsd:element name="UpdateChannel"
            type="typens:UpdateChannel" minOccurs="0" />
          <xsd:element name="CreateRole" type="typens:CreateRole"
            minOccurs="0" />
          <xsd:element name="DeleteRole" type="typens:DeleteRole"
            minOccurs="0" />
          <xsd:element name="UpdateRole" type="typens:UpdateRole"
            minOccurs="0" />
          <xsd:element name="CreateFileType"
            type="typens:CreateFileType" minOccurs="0" />
           <xsd:element name="DeleteFileType"
            type="typens:DeleteFileType" minOccurs="0" />
          <xsd:element name="UpdateFileType"
            type="typens:UpdateFileType" minOccurs="0" />
          <xsd:element name="CreateFolder"
            type="typens:CreateFolder"
            minOccurs="0" />
          <xsd:element name="DeleteFile" type="typens:DeleteFile"
            minOccurs="0" />
          <xsd:element name="MoveFile" type="typens:MoveFile"
            minOccurs="0" />
          <xsd:element name="CopyFile" type="typens:CopyFile"
            minOccurs="0" />
          <xsd:element name="UpdateFile" type="typens:UpdateFile"
            minOccurs="0" />
          <xsd:element name="DeleteJob" type="typens:DeleteJob"
            minOccurs="0" />
          <xsd:element name="DeleteJobNotices"
            type="typens:DeleteJobNotices" minOccurs="0" />
          <xsd:element name="UpdateJobSchedule"
          <xsd:element name="DeleteJobSchedule"
            type="typens:DeleteJobSchedule" minOccurs="0"/>
            type="typens:UpdateJobSchedule" minOccurs="0" />
          <xsd:element name="UpdateVolumeProperties"
            type="typens:UpdateVolumeProperties" minOccurs="0" />
           <xsd:element name="UpdateOpenSecurityCache"
            type="typens:UpdateOpenSecurityCache" minOccurs="0" />
        </xsd:choice>
      </xsd:sequence>
    </xsd:complexType>
```

Request elements

CreateUser
Creates a user in the Encyclopedia volume.

DeleteUser
Deletes one or more users.

UndeleteUser
Undeletes one or more users.

UpdateUser
Updates user properties in the Encyclopedia volume.

CreateGroup
Creates a notification group in an Encyclopedia volume.

DeleteGroup
Deletes one or more notification groups.

UpdateGroup
Updates notification group properties in the Encyclopedia volume.

CreateChannel
Creates a channel in an Encyclopedia volume.

DeleteChannel
Deletes channels from the Encyclopedia volume.

UpdateChannel
Updates channel properties in the Encyclopedia volume.

CreateRole
Creates a security role in the Encyclopedia volume.

DeleteRole
Deletes one or more security roles.

UpdateRole
Updates security role properties in the Encyclopedia volume.

CreateFileType
Creates a new file type in BIRT iHub.

DeleteFileType
Deletes file types.

UpdateFileType
Updates file type properties in the Encyclopedia volume.

CreateFolder
Creates a folder in an Encyclopedia volume.

DeleteFile
Deletes files or folders from the Encyclopedia volume.

MoveFile
Moves a file or folder from the working directory to a specified target directory in the Encyclopedia volume.

CopyFile
Copies a file or folder in the working directory to a specified target directory.

UpdateFile
Updates file or folder properties in the Encyclopedia volume.

DeleteJob
Deletes one or more jobs.

DeleteJobNotices
Deletes one or more job notices.

DeleteJobSchedule
Updates a job schedule.

UpdateJobSchedule
Updates a job schedule.

UpdateVolumeProperties
Updates the properties of a specific Encyclopedia volume.

UpdateOpenSecurityCache
Flushes the Encyclopedia volume's open security cache and retrieves new data from an external security source.

UndeleteUser

Undeletes users, reversing a DeleteUser operation within the unit of work of an AdminOperation. To undelete a single user, specify Id. To delete several users, specify IdList.

Request schema

```
<xsd:complexType name="UndeleteUser">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="Name" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="IgnoreMissing" type="xsd:boolean"
         minOccurs="0"/>
      </xsd:sequence>
</xsd:complexType>
```

Request elements

IdList
The list of user IDs to undelete.

Id
The ID of the single user to undelete.

IgnoreMissing
Specifies what to do if the specified user does not exist. If True, BIRT iHub ignores the request. If False, the operation reports an error and stops. The default value is True.

UpdateChannel

Updates channel properties. To update channel properties, specify the types of updates to make using UpdateChannelOperationGroup, then specify which channels to update.

To update a single channel, specify Name or Id. To update a list of channels, specify NameList or IdList. To update channels matching the specified conditions, specify Search.

Request schema

```
<xsd:complexType name="UpdateChannel">
   <xsd:sequence>
      <xsd:element name="UpdateChannelOperationGroup"
         type="typens:UpdateChannelOperationGroup/>
      <xsd:choice>
         <xsd:element name="Search" type="typens:ChannelSearch"/>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="NameList" type="typens:ArrayOfString"/>
         <xsd:element name="Id" type="xsd:string"/>
         <xsd:element name="Name" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="IgnoreMissing" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="IgnoreDup" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="SkipPermissionError" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UpdateChannelOperationGroup
The tasks to perform.

Search
The search condition that specifies which channels to update.

IdList
The list of channel IDs to update. Specify either IdList or NameList.

NameList
The list of channel names to update. Specify either NameList or IdList.

Id
The ID of the single channel to update. Specify either Id or Name.

Name
The name of the single channel to update. Specify either Name or Id.

IgnoreMissing
Specifies what to do if the specified channel does not exist. If True, BIRT iHub continues the operation. If False, the operation reports an error and stops. The default value is True.

IgnoreDup
Specifies whether to report an error for a duplicate request. BIRT iHub always rejects a duplicate request regardless of the IgnoreDup setting. If True, BIRT iHub does not report an error. In a name list or ID list, True prevents BIRT iHub from performing duplicate operations. If False, BIRT iHub reports an error. The default value is False.

SkipPermissionError
Specifies whether to continue the request if BIRT iHub produces a permission error. BIRT iHub produces this error if the user you are subscribing to the channel does not have read and write privileges to the channel. If True, the BIRT iHub ignores the error.

UpdateChannelOperation

Specifies the tasks to perform during the UpdateChannel operation.

Request schema

```
<xsd:complexType name="UpdateChannelOperation">
   <xsd:sequence>
      <xsd:element name="SetAttributes" type="typens:Channel"
         minOccurs="0"/>
      <xsd:element name="AddSubscribersByName"
         type="typens:ArrayOfString" minOccurs="0"/>
      <xsd:element name="RemoveSubscribersByName"
         type="typens:ArrayOfString" minOccurs="0"/>
      <xsd:element name="SetSubscribersByName"
         type="typens:ArrayOfString" minOccurs="0"/>
      <xsd:element name="AddSubscribersById"
         type="typens:ArrayOfString"
         minOccurs="0"/>
```

```
      <xsd:element name="RemoveSubscribersById"
         type="typens:ArrayOfString" minOccurs="0"/>
      <xsd:element name="SetSubscribersById"
         type="typens:ArrayOfString"
         minOccurs="0"/>
      <xsd:element name="GrantPermissions"
         type="typens:ArrayOfPermission" minOccurs="0"/>
      <xsd:element name="RevokePermissions"
         type="typens:ArrayOfPermission" minOccurs="0"/>
      <xsd:element name="SetPermissions"
         type="typens:ArrayOfPermission"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

SetAttributes
The general attributes to update.

AddSubscribersByName
The names of users to subscribe to the channel. Specify either AddSubscribersByName or AddSubscribersById.

RemoveSubscribersByName
The names of users to unsubscribe from the channel. Specify either RemoveSubscribersByName or RemoveSubscribersById.

SetSubscribersByName
The names of users for whom to update channel subscription. Specify either SetSubscribersByName or SetSubscribersById.

AddSubscribersById
The IDs of users to subscribe to the channel. Specify either AddSubscribersById or AddSubscribersByName.

RemoveSubscribersById
The IDs of users to unsubscribe from the channel. Specify either RemoveSubscribersById or RemoveSubscribersByName.

SetSubscribersById
The IDs of users for whom to update channel subscription. Specify either SetSubscribersById or SetSubscribersByName.

GrantPermissions
Adds an access control list (ACL) to the channel.

RevokePermissions
Removes an ACL from the channel.

SetPermissions
Updates the ACL for the channel.

UpdateChannelOperationGroup

Specifies the UpdateChannelOperation element used within the UpdateChannel operation.

Request schema

```
<xsd:complexType name="UpdateChannelOperationGroup">
   <xsd:sequence>
      <xsd:element name="UpdateChannelOperation"
         type="typens:UpdateChannelOperation" maxOccurs="unbounded"
         minOccurs="0"/>
   </xsd:sequence>
<xsd:complexType>
```

Request elements

UpdateChannelOperation
The UpdateChannelOperation element for the group.

UpdateDatabaseConnection

Updates an Actuate Caching service (ACS) database connection.

Request schema

```
<xsd:complexType name="UpdateDatabaseConnection">
   <xsd:sequence>
      <xsd:element name="DatabaseConnection"
         type="typens:DatabaseConnectionDefinition"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

DatabaseConnection
The information to update.

Response schema

```
<xsd:complexType name="UpdateDatabaseConnectionResponse">
   <xsd:sequence>
      <xsd:element name="FileId" type="xsd:string"/>
      <xsd:element name="Warnings" minOccurs="0"
         type="typens:ArrayOfString"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

FileId
The ID of the updated connection object.

Warnings
Any problems that occur when iHub attempts to update the ACS database connection.

UpdateFile

Updates files or folders. To update files or folders, specify the types of updates to make using UpdateFileOperationGroup, then specify which files or folders to update.

To update the properties of a file or folder, you must have the write privilege on the file or folder. To update privileges to the file or folder, you must have the grant privilege on the file or folder.

To update a single file or folder, specify Name or Id. To update a list of files or folders, specify NameList or IdList. To update files or folders matching the specified conditions, specify Search.

Request schema

```
<xsd:complexType name="UpdateFile">
   <xsd:sequence>
      <xsd:choice minOccurs="0">
         <xsd:element name="WorkingFolderId" type="xsd:string"/>
         <xsd:element name="WorkingFolderName" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="LatestVersionOnly" type="xsd:boolean"
            minOccurs="0"/>
      <xsd:element name="Recursive" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="UpdateFileOperationGroup"/>
      <xsd:choice>
         <xsd:element name="Search" type="typens:FileSearch"/>
         <xsd:element name="NameList" type="typens:ArrayOfString"/>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="Id" type="xsd:string"/>
         <xsd:element name="Name" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="IgnoreMissing" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

WorkingFolderId
The ID of the working folder of the file or folder to update. Specify either WorkingFolderId or WorkingFolderName.

WorkingFolderName
The name of the working folder of the file or folder to update. Specify either WorkingFolderName or WorkingFolderId.

LatestVersionOnly
Specifies whether to search all versions of the file. If True, the search includes only the latest version. The default value is False.

Recursive
Specifies whether to search subfolders. If True, the search includes subfolders. The default value is False.

UpdateFileOperationGroup
The tasks to perform.

Search
The search conditions. If conditions apply to multiple fields, use ConditionArray.

NameList
The list of file or folder names to update. Specify either NameList or IdList.

IdList
The list of file or folder IDs to update. Specify either IdList or NameList.

Id
The ID of the single file or folder to update. Specify either Id or Name.

Name
The name of the single file or folder to update. Specify either Name or Id.

IgnoreMissing
Specifies what to do if the specified file or folder does not exist. If True, BIRT iHub ignores the request. If False, the operation reports an error and stops. The default value is True.

UpdateFileOperation

Specifies the tasks to perform during the UpdateFile operation. To specify which files to update, use UpdateFile.

Request schema

```
<xsd:complexType name="UpdateFileOperationGroup">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="SetAttributes" type="typens:File"
            minOccurs="0"/>
         <xsd:element name="AddDependentFilesByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="RemoveDependentFilesByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="SetDependentFilesByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="AddRequiredFilesByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="RemoveRequiredFilesByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="SetRequiredFilesByName"
```

```
            type="typens:ArrayOfString" minOccurs="0"/>
          <xsd:element name="AddDependentFilesById"
            type="typens:ArrayOfString" minOccurs="0"/>
          <xsd:element name="RemoveDependentFilesById"
            type="typens:ArrayOfString" minOccurs="0"/>
          <xsd:element name="SetDependentFilesById"
            type="typens:ArrayOfString" minOccurs="0"/>
          <xsd:element name="AddRequiredFilesById"
            type="typens:ArrayOfString" minOccurs="0"/>
          <xsd:element name="RemoveRequiredFilesById"
            type="typens:ArrayOfString" minOccurs="0"/>
          <xsd:element name="SetRequiredFilesById"
            type="typens:ArrayOfString" minOccurs="0"/>
          <xsd:element name="GrantPermissions"
            type="typens:ArrayOfPermission" minOccurs="0"/>
          <xsd:element name="RevokePermissions"
            type="typens:ArrayOfPermission" minOccurs="0"/>
          <xsd:element name="SetPermissions"
            type="typens:ArrayOfPermission" minOccurs="0"/>
          <xsd:element name="AddArchiveRules"
            type="typens:ArrayOfArchiveRule" minOccurs="0"/>
          <xsd:element name="RemoveArchiveRules"
            type="typens:ArrayOfString" minOccurs="0"/>
          <xsd:element name="SetArchiveRules"
            type="typens:ArrayOfArchiveRule" minOccurs="0"/>
          <xsd:element name="SetParameterDefinitions"
            type="typens:ArrayOfParameterDefinition" minOccurs="0"/>
        </xsd:choice>
      </xsd:sequence>
    </xsd:complexType>
```

Request elements

SetAttributes
The general attributes to update.

AddDependentFilesByName
The names of files to add as dependents. Specify either AddDependentFilesByName or AddDependentFilesById.

RemoveDependentFilesByName
The names of dependent files to remove. Specify either RemoveDependentFilesByName or RemoveDependentFilesById.

SetDependentFilesByName
The names of dependent files to update. Specify either SetDependentFilesByName or SetDependentFilesById.

AddRequiredFilesByName
The names of required files to add. Specify either AddRequiredFilesByName or AddRequiredFilesById.

RemoveRequiredFilesByName
The names of required files to remove. Specify either RemoveRequiredFilesByName or RemoveRequiredFilesById.

SetRequiredFilesByName
The names of required files to update. Specify either SetRequiredFilesByName or SetRequiredFilesById.

AddDependentFilesById
The IDs of files to add as dependents. Specify either AddDependentFilesById or AddDependentFilesByName.

RemoveDependentFilesById
The IDs of dependent files to remove. Specify either RemoveDependentFilesById or RemoveDependentFilesByName.

SetDependentFilesById
The IDs of dependent files to update. Specify either SetDependentFilesById or SetDependentFilesByName.

AddRequiredFilesById
The IDs of required files to add. Specify either AddRequiredFilesById or AddRequiredFilesByName.

RemoveRequiredFilesById
The IDs of required files to remove. Specify either RemoveRequiredFilesById or RemoveRequiredFilesByName.

SetRequiredFilesById
The IDs of required files to update. Specify either SetRequiredFilesById or SetRequiredFilesByName.

GrantPermissions
The new privileges to grant. You cannot grant privileges to a file with private access.

RevokePermissions
The privileges to revoke. You cannot revoke privileges to a file with private access.

SetPermissions
The privileges to update. You cannot update privileges to a file with private access.

AddArchiveRules
The new autoarchive rules.

RemoveArchiveRules
The autoarchive rules to remove.

SetArchiveRules
The autoarchive rules to update.

SetParameterDefinitions
The dynamic report parameters for third-party executable files.

UpdateFileOperationGroup

Specifies the UpdateFileOperation element within UpdateFile.

Request schema

```
<xsd:complexType name="UpdateFileOperationGroup">
   <xsd:sequence>
      <xsd:element name="UpdateFileOperation"
         type="typens:UpdateFileOperation"
         maxOccurs="unbounded" minOccurs="0"/>
      </xsd:sequence>
</xsd:complexType>
```

Request elements

UpdateFileOperation
The UpdateFileOperation element to use during the UpdateFile operation.

UpdateFileType

Updates file types. To update file types, specify the types of updates to make using UpdateFileTypeOperationGroup, then specify which file types to update.

To update a single file type, specify Name or Id. To update a list of file types, specify NameList or IdList.

Request schema

```
<xsd:complexType name="UpdateFileType">
   <xsd:sequence>
      <xsd:element name="UpdateFileTypeOperationGroup"
         type="typens:UpdateFileTypeOperationGroup"/>
      <xsd:choice>
         <xsd:element name="NameList" type="typens:ArrayOfString"/>
         <xsd:element name="Name" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="IgnoreMissing" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="IgnoreDup" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UpdateFileTypeOperationGroup
The tasks to perform.

NameList
The list of file types to update.

Name
The name of a single file type to update.

IgnoreMissing
Specifies what to do if the specified file type does not exist. If True, BIRT iHub ignores the request. If False, the operation reports an error and stops. The default value is True.

IgnoreDup
Specifies whether to report an error for a duplicate request. BIRT iHub always rejects a duplicate request regardless of the IgnoreDup setting. If True, BIRT iHub does not report an error. In a name list or ID list, True prevents BIRT iHub from performing duplicate operations. If False, BIRT iHub reports an error. The default value is False.

UpdateFileTypeOperation

Specifies the tasks to perform during the UpdateFileType operation.

Request schema

```
<xsd:complexType name="UpdateFileTypeOperation">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="SetAttributes" type="typens:FileType"
            minOccurs="0"/>
         <xsd:element name="SetParameterDefinitions"
            type="typens:ArrayOfParameterDefinition" minOccurs="0"/>
         <xsd:element name="SetWindowsIcon" type="xsd:base64Binary"
            minOccurs="0"/>
         <xsd:element name="SetLargeWebIcon"
            type="xsd:base64Binary" minOccurs="0"/>
         <xsd:element name="SetSmallWebIcon"
            type="xsd:base64Binary" minOccurs="0"/>
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

SetAttributes
The general attributes to update.

SetParameterDefinitions
The parameters to update.

SetWindowsIcon
The Windows icon to display for the file type.

SetLargeWebIcon
The large icon to display for the file type in a browser.

SetSmallWebIcon
The small icon to display for the file type in a browser.

UpdateFileTypeOperationGroup

Specifies the UpdateFileTypeOperation element to use during the UpdateFileType operation.

Request schema

```
<xsd:complexType name="UpdateFileTypeOperationGroup">
   <xsd:sequence>
      <xsd:element name="UpdateFileTypeOperation"
         type="typens:UpdateFileTypeOperation"
            maxOccurs="unbounded"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UpdateFileOperation
The UpdateFileOperation element to use in the UpdateFileType operation.

UpdateGroup

Updates notification groups. To update groups, specify the types of updates to make using UpdateGroupOperationGroup, then specify which groups to update.

To update a single group, specify Name or Id. To update a list of groups, specify NameList or IdList. To update groups matching the specified conditions, specify Search.

Request schema

```
<xsd:complexType name="UpdateGroup">
   <xsd:sequence>
      <xsd:element name="UpdateGroupOperationGroup"
         type="typens:UpdateGroupOperationGroup"/>
      <xsd:choice>
         <xsd:element name="Search" type="typens:GroupSearch"/>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="NameList" type="typens:ArrayOfString"/>
         <xsd:element name="Id" type="xsd:string"/>
         <xsd:element name="Name" type="xsd:string"/>
```

```
      </xsd:choice>
      <xsd:element name="IgnoreMissing" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="IgnoreDup" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UpdateGroupOperationGroup
The tasks to perform.

Search
The search condition that specifies which groups to update.

IdList
The list of group IDs to update. Specify either IdList or NameList.

NameList
The list of group names to update. Specify either NameList or IdList.

Id
The ID of the single group to update. Specify either Id or Name.

Name
The name of the single group to update. Specify either Name or Id.

IgnoreMissing
Specifies what to do if the specified group does not exist. If True, BIRT iHub ignores the request. If False, the operation reports an error and stops. The default value is True.

IgnoreDup
Specifies whether to report an error for a duplicate request. BIRT iHub always rejects a duplicate request regardless of the IgnoreDup setting. If True, BIRT iHub does not report an error. In a name list or ID list, True prevents BIRT iHub from performing duplicate operations. If False, BIRT iHub reports an error. The default value is False.

UpdateGroupOperation

Specifies the tasks to perform during the UpdateGroup operation.

Request schema

```
<xsd:complexType name="UpdateGroupOperation">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="SetAttributes" type="typens:Group"
            minOccurs="0"/><xsd:element name="AddUsersByName"
            type="typens:ArrayOfString"minOccurs="0"/>
```

```
            <xsd:element name="RemoveUsersByName"
               type="typens:ArrayOfString" minOccurs="0"/>
            <xsd:element name="SetUsersByName"
               type="typens:ArrayOfString"minOccurs="0"/>
            <xsd:element name="AddUsersById"
               type="typens:ArrayOfString"
               minOccurs="0"/>
            <xsd:element name="RemoveUsersById"
               type="typens:ArrayOfString" minOccurs="0"/>
            <xsd:element name="SetUsersById"
               type="typens:ArrayOfString"
               minOccurs="0"/>
         </xsd:choice>
      </xsd:sequence>
   </xsd:complexType>
```

Request elements

SetAttributes
The general attributes to update.

AddUsersByName
Adds the specified user names to the group. Specify either AddUsersByName or AddUsersById.

RemoveUsersByName
Removes the specified user names from the group. Specify either RemoveUsersByName or RemoveUsersById.

SetUsersByName
Updates the specified users' group membership. Specify either SetUsersByName or SetUsersById.

AddUsersById
Adds the specified user IDs to the group. Specify either AddUsersById or AddUsersByName.

RemoveUsersById
Removes the specified user IDs form the group. Specify either RemoveUsersById or RemoveUsersByName.

SetUsersById
Updates the specified users' group membership. Specify either SetUsersById or SetUsersByName.

UpdateGroupOperationGroup

Specifies the UpdateGroupOperation element to use during the UpdateGroup operation.

Request schema

```
<xsd:complexType name="UpdateGroupOperationGroup">
   <xsd:sequence>
      <xsd:element name="UpdateGroupOperation"
         type="typens:UpdateGroupOperation" maxOccurs="unbounded"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UpdateGroupOperation
The UpdateGroupOperation element for use with the UpdateGroup operation.

UpdateJobSchedule

Updates job schedules. To update scheduled jobs, specify the types of updates to make using UpdateJobScheduleOperationGroup, then specify which jobs to update.

To update a single scheduled job, specify Id. To update a list of scheduled jobs, specify IdList. To update scheduled jobs matching the specified conditions, specify Search.

Request schema

```
<xsd:complexType name="UpdateJobSchedule">
   <xsd:sequence>
      <xsd:element name="UpdateJobScheduleOperationGroup"
         type="typens:UpdateJobScheduleOperationGroup"/>
      <xsd:choice>
         <xsd:element name="Search"
            type="typens:JobScheduleSearch"/>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="Id" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="IgnoreMissing" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UpdateJobScheduleOperationGroup
The tasks to perform.

Search
The search conditions. If conditions apply to multiple fields, use ConditionArray.

IdList
The list of job IDs to update.

Id
The ID of a single job to update.

IgnoreMissing
Specifies what to do if the specified job does not exist. If True, BIRT iHub ignores the request. If False, the operation reports an error and stops. The default value is True.

UpdateJobScheduleOperation

Specifies the tasks to perform during the UpdateJobSchedule operation.

Request schema

```
<xsd:complexType name="UpdateJobScheduleOperation">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="SetAttributes"
            ype="typens:JobProperties" minOccurs="0"/>
         <xsd:element name="SetParameters"
            type="typens:JobInputDetail" minOccurs="0"/>
         <xsd:element name="SetPrinterOptions"
            type="typens:JobPrinterOptions" minOccurs="0"/>
         <xsd:element name="SetSchedules" type="typens:JobSchedule"
            minOccurs="0"/>
         <xsd:element name="AddUserNotificationByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="RemoveUserNotificationByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="SetUserNotificationByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="AddGroupNotificationByName"
            type="typens:ArrayOfString" minOccurs="0"/>
            <xsd:element name="RemoveGroupNotificationByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="SetGroupNotificationByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="AddChannelNotificationByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="RemoveChannelNotificationByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="SetChannelNotificationByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="AddUserNotificationById"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="RemoveUserNotificationById"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="SetUserNotificationById"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="AddGroupNotificationById"
            type="typens:ArrayOfString" minOccurs="0"/>
```

```
            <xsd:element name="RemoveGroupNotificationById"
              type="typens:ArrayOfString" minOccurs="0"/>
            <xsd:element name="SetGroupNotificationById"
              type="typens:ArrayOfString" minOccurs="0"/>
            <xsd:element name="AddChannelNotificationById"
              type="typens:ArrayOfString" minOccurs="0"/>
            <xsd:element name="RemoveChannelNotificationById"
              type="typens:ArrayOfString" minOccurs="0"/>
            <xsd:element name="SetChannelNotificationById"
              type="typens:ArrayOfString" minOccurs="0"/>
            <xsd:element name="AddOutputFilePermissions"
              type="typens:ArrayOfPermission" minOccurs="0"/>
            <xsd:element name="RemoveOutputFilePermissions"
              type="typens:ArrayOfPermission" minOccurs="0"/>
            <xsd:element name="SetOutputFilePermissions"
              type="typens:ArrayOfPermission" minOccurs="0"/>
            <xsd:element name="SetParameterValues"
              type="typens:ArrayOfParameterValue" minOccurs="0"/>
            <xsd:element name="SetQuery" type="typens:Query"
              minOccurs="0"/>
            <xsd:element name="SetOutputFileAccess"
              type="typens:FileAccess"minOccurs="0"/>
            <xsd:element name="SetWaitForEvent" type="typens:Event"
              minOccurs="0"/>
          </xsd:choice>
        </xsd:sequence>
      </xsd:complexType>
```

Request elements

SetAttributes
The general attributes to update.

SetParameters
The input parameters, output parameters, autoarchive settings, and distribution location settings to update.

SetPrinterOptions
The printer options to update.

SetSchedules
The schedules to set.

AddUserNotificationByName
The name of the user to add to the notification list. Specify either AddUserNotificationByName or AddUserNotificationById.

RemoveUserNotificationByName
The name of the user to remove from the notification list. Specify either RemoveUserNotificationByName or RemoveUserNotificationById.

SetUserNotificationByName
The name of the user for whom to update notification. Specify either SetUserNotificationByName or SetUserNotificationById.

AddGroupNotificationByName
The name of the group to add to the notification list. Specify either AddGroupNotificationByName or AddGroupNotificationById.

RemoveGroupNotificationByName
The name of the group to remove from the notification list. Specify either RemoveGroupNotificationByName or RemoveGroupNotificationById.

SetGroupNotificationByName
The name of the group for which to update notification. Specify either SetGroupNotificationByName or SetGroupNotificationById.

AddChannelNotificationByName
The name of the channel to add to the notification list. Specify either AddChannelNotificationByName or AddUserNotificationById.

RemoveChannelNotificationByName
The name of the channel to remove from the notification list. Specify either RemoveChannelNotificationByName or RemoveChannelNotificationById.

SetChannelNotificationByName
The name of the channel for which to update notification. Specify either SetChannelNotificationByName or SetChannelNotificationById.

AddUserNotificationById
The ID of the user to add to the notification list. Specify either AddUserNotificationById or AddUserNotificationByName.

RemoveUserNotificationById
The ID of the user to remove from the notification list. Specify either RemoveUserNotificationById or RemoveUserNotificationByName.

SetUserNotificationById
The ID of the user for whom to update notification. Specify either SetUserNotificationById or SetUserNotificationByName.

AddGroupNotificationById
The ID of the group to add to the notification list. Specify either AddGroupNotificationById or AddGroupNotificationByName.

RemoveGroupNotificationById
The ID of the group to remove from the notification list. Specify either RemoveGroupNotificationById or RemoveGroupNotificationByName.

SetGroupNotificationById
The ID of the group for which to update notification. Specify either SetGroupNotificationById or SetGroupNotificationByName.

AddChannelNotificationById
The ID of the channel to add to the notification list. Specify either AddChannelNotificationById or AddUserNotificationByName.

RemoveChannelNotificationById
The ID of the channel to remove from the notification list. Specify either RemoveChannelNotificationById or RemoveChannelNotificationByName.

SetChannelNotificationById
The ID of the channel for which to update notification. Specify either SetChannelNotificationById or SetChannelNotificationByName.

AddOutputFilePermissions
The output file permissions to add. You cannot add file permissions to a file with private access.

RemoveOutputFilePermissions
The output file permissions to remove. You cannot remove file permissions from a file with private access.

SetOutputFilePermissions
The output file permissions to update. You cannot update file permissions of a file with private access.

SetParameterValues
The parameter values to update.

SetQuery
The query parameters to update.

SetOutputFileAccess
The access rights to the output file to update. Valid values are:

- Private
  Only the owner of the file and an administrator can access the file.
- Shared
  All users and roles specified in the access control list (ACL) for the file can access the file.

SetWaitForEvent
The event to set that is being waited on. When set, processes that are waiting for this event will proceed.

UpdateJobScheduleOperationGroup

Specifies the UpdateJobScheduleOperation element within the UpdateJobSchedule operation.

Request schema

```
<xsd:complexType name="UpdateJobScheduleOperationGroup">
   <xsd:sequence>
      <xsd:element name="UpdateJobScheduleOperation"
         type="typens:UpdateJobScheduleOperation"
         maxOccurs="unbounded" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UpdateJobScheduleOperation
The UpdateJobScheduleOperation element for use with the UpdateJobSchedule operation.

UpdateOpenSecurityCache

Flushes the Encyclopedia volume's open security cache and retrieves new data from the external security source. Use UpdateOpenSecurityCache when information in the external data source has changed and must be updated immediately.

In the request, specify the list of users, roles, groups, and translated user or role names to update. If no users, roles, groups, or translated user and role names are specified, all users, roles, groups, and translated user and role names are updated. Otherwise, only the specified items are updated.

Request schema

```
<xsd:complexType name="UpdateOpenSecurityCache">
   <xsd:sequence>
      <xsd:element name="UserNameList"
         type="actu:ArrayOfString" minOccurs="0"/>
      <xsd:element name="GroupList"
         type="actu:ArrayOfString" minOccurs="0"/>
      <xsd:element name="RoleList"
         type="actu:ArrayOfString" minOccurs="0"/>
      <xsd:element name="TranslatedUserNames"
         type="typens:ExternalTranslatedUserNames" minOccurs="0"/>
      <xsd:element name="TranslatedRoleNames"
         type="typens:ExternalTranslatedRoleNames" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UserNameList
The list of user names to retrieve.

GroupList
The list of groups to retrieve.

RoleList
The list of roles to retrieve.

TranslatedUserNames
The translated user names to retrieve, such as Administrator.

TranslatedRoleNames
The translated role names to retrieve, such as Administrator, Operator, and All.

UpdateResourceGroup

Updates resource group properties. You cannot update the resource group name, type, or the name of the BIRT iHub on which the resource group runs.

Request schema

```
<xsd:complexType name="UpdateResourceGroup">
   <xsd:sequence>
      <xsd:element name="ResourceGroup"
         type="typens:ResourceGroup"/>
      <xsd:element name="ResourceGroupSettingsList"
         type="typens:ArrayOfResourceGroupSettings" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ResourceGroup
Contains one or more of the following properties to update:

- Disabled
- Description
- MinPriority
- MaxPriority
- Reserved

ResourceGroupSettingsList
Contains one or more of the following properties to update:

- Activate
- MaxFactory
- FileTypes

Response schema

```
<xsd:complexType name="UpdateResourceGroupResponse">
   <xsd:sequence/>
</xsd:complexType>
```

UpdateRole

Updates roles. To update roles, specify the types of updates to make using UpdateRoleOperationGroup, then specify which roles to update.

To update a single role, specify Name or Id. To update a list of roles, specify NameList or IdList. To update roles matching the specified conditions, specify Search.

Request schema

```
<xsd:complexType name="UpdateRole">
   <xsd:sequence>
      <xsd:element name="UpdateRoleOperationGroup"
         type="typens:UpdateRoleOperationGroup"/>
         <xsd:choice>
            <xsd:element name="Search" type="typens:RoleSearch"/>
            <xsd:element name="IdList" type="typens:ArrayOfString"/>
            <xsd:element name="NameList"
               type="typens:ArrayOfString"/>
            <xsd:element name="Id" type="xsd:string"/>
            <xsd:element name="Name" type="xsd:string"/>
         </xsd:choice>
         <xsd:element name="IgnoreMissing" type="xsd:boolean"
            minOccurs="0"/>
         <xsd:element name="IgnoreDup" type="xsd:boolean"
            minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UpdateRoleOperationGroup
The tasks to perform.

Search
The search condition that specifies which roles to update.

IdList
The list of role IDs to update. Specify either IdList or NameList.

NameList
The list of role names to update. Specify either NameList or IdList.

Id
The ID of the single role to update. Specify either Id or Name.

Name
The name of the single role to update. Specify either Name or Id.

IgnoreMissing
Specifies what to do if the specified role does not exist. If True, BIRT iHub ignores the request. If False, the operation reports an error and stops. The default value is True.

IgnoreDup
Specifies whether to report an error for a duplicate request. BIRT iHub always rejects a duplicate request regardless of the IgnoreDup setting. If True, BIRT iHub does not report an error. In a name list or ID list, True prevents BIRT iHub from performing duplicate operations. If False, BIRT iHub reports an error. The default value is False.

UpdateRoleOperation

Specifies the tasks to perform during the UpdateRole operation.

Request schema

```
<xsd:complexType name="UpdateRoleOperation">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="SetAttributes" type="typens:Role"
            minOccurs="0"/>
         <xsd:element name="AssignedToUsersByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="DroppedFromUsersByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="SetBearingUsersByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="AddChildRolesByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="RemoveChildRolesByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="SetChildRolesByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="AddParentRolesByName"
         type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="RemoveParentRolesByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="SetParentRolesByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="AssignedToUsersById"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="DroppedFromUsersById"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="SetBearingUsersById"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="AddChildRolesById"
```

```
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="RemoveChildRolesById"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="SetChildRolesById"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="AddParentRolesById"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="RemoveParentRolesById"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="SetParentRolesById"
         type="typens:ArrayOfString" minOccurs="0"/>
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

SetAttributes
The general attributes to update.

AssignedToUsersByName
The names of roles to which to assign users. Specify either AssignedToUsersByName or AssignedToUsersById.

DroppedFromUsersByName
The names of roles from which to delete users. Specify either DroppedFromUsersByName or DroppedFromUsersById.

SetBearingUsersByName
The names of roles to which to reassign users. Specify either SetBearingUsersByName or SetBearingUsersById.

AddChildRolesByName
The names of roles to add to the role as descendant roles. Specify either AddChildRolesByName or AddChildRolesById.

RemoveChildRolesByName
The names of descendant roles to remove from the role. Specify either RemoveChildRolesByName or RemoveChildRolesById.

SetChildRolesByName
The names of the role's descendant roles. Specify either SetChildRolesByName or SetChildRolesById.

AddParentRolesByName
The names of roles to add to the role as its ascendant roles. Specify either AddParentRolesByName or AddParentRolesById.

RemoveParentRolesByName
The name of the parent role to remove. Specify either RemoveParentRolesByName or RemoveParentRolesById.

SetParentRolesByName
The names of the role's ascendant roles. Specify either SetParentRolesByName or SetParentRolesById.

AssignedToUsersById
The IDs of roles to which to assign users. Specify either AssignedToUsersById or AssignedToUsersByName.

DroppedFromUsersById
The IDs of roles from which to delete users. Specify either DroppedFromUsersById or DroppedFromUsersByName.

SetBearingUsersById
The IDs of roles to which to reassign users. Specify either SetBearingUsersById or SetBearingUsersByName.

AddChildRolesById
The IDs of roles to add to the role as descendant roles. Specify either AddChildRolesByName or AddChildRolesById.

RemoveChildRolesById
The IDs of descendant roles to remove from the role. Specify either RemoveChildRolesById or RemoveChildRolesByName.

SetChildRolesById
The IDs of the role's descendant roles. Specify either SetChildRolesById or SetChildRolesByName.

AddParentRolesById
The IDs of roles to add to the role as its ascendant roles. Specify either AddParentRolesById or AddParentRolesByName.

RemoveParentRolesById
The IDs of the ascendant roles to remove. Specify either RemoveParentRolesByName or RemoveParentRolesById.

SetParentRolesById
The IDs of the role's ascendant roles. Specify either SetParentRolesById or SetParentRolesByName.

UpdateRoleOperationGroup

Specifies the UpdateRoleOperation element to use during the UpdateRole operation.

Request schema

```
<xsd:complexType name="UpdateRoleOperationGroup">
  <xsd:sequence>
```

```
   <xsd:element name="UpdateRoleOperation"
        type="typens:UpdateRoleOperation"
        maxOccurs="unbounded" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UpdateRoleOperation
The UpdateRoleOperation element for use with the UpdateRole operation.

UpdateUser

Updates user properties. To update users, specify the types of updates to make using UpdateUserOperationGroup, then specify which users to update.

To update a single user, specify Name or Id. To update a list of users, specify NameList or IdList. To update users matching the specified conditions, specify Search.

Request schema

```
<xsd:complexType name="UpdateUser">
   <xsd:sequence>
      <xsd:element name="UpdateUserOperationGroup"
         type="typens:UpdateUserOperationGroup"/>
      <xsd:choice>
         <xsd:element name="Search" type="typens:UserSearch"/>
         <xsd:element name="IdList" type="typens:ArrayOfString"/>
         <xsd:element name="NameList" type="typens:ArrayOfString"/>
         <xsd:element name="Id" type="xsd:string"/>
         <xsd:element name="Name" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="IgnoreMissing" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="IgnoreDup" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="SkipPermissionError" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UpdateUserOperationGroup
The tasks to perform.

Search
The search conditions. If search conditions apply to multiple fields, use ConditionArray.

IdList
The list of user IDs to update. Specify either IdList or NameList.

NameList
The list of user names to update. Specify either NameList or IdList.

Id
The ID of a single user to update. Specify either Id or Name.

Name
The name of a single user to update. Specify either Name or Id.

IgnoreMissing
Specifies what to do if the specified user does not exist. If True, BIRT iHub ignores the request. If False, the operation reports an error and stops. The default value is True.

IgnoreDup
Specifies whether to report an error for a duplicate request. BIRT iHub always rejects a duplicate request regardless of the IgnoreDup setting. If True, BIRT iHub does not report an error. In a name list or ID list, True prevents BIRT iHub from performing duplicate operations. If False, BIRT iHub reports an error. The default value is False.

SkipPermissionError
Specifies whether to continue the request if BIRT iHub produces a permission error. BIRT iHub produces this error if the user you are subscribing to a channel does not have read and write privileges to the channel. If True, the BIRT iHub ignores the error.

UpdateUserOperation

Specifies the tasks to perform during the UpdateUser operation.

Request schema

```
<xsd:complexType name="UpdateUserOperation">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="SetAttributes" type="typens:User"
            minOccurs="0"/>
         <xsd:element name="AddToGroupsByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="RemoveFromGroupsByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="SetGroupMembershipsByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="AssignRolesByName"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="DropRolesByName"
            type="typens:ArrayOfString"
            minOccurs="0"/>
```

```
          <xsd:element name="SetRolesByName"
            type="typens:ArrayOfString"
            minOccurs="0"/>
          <xsd:element name="SubscribeToChannelsByName"
            type="typens:ArrayOfString" minOccurs="0"/>
          <xsd:element name="UnsubscribeFromChannelsByName"
            type="typens:ArrayOfString" minOccurs="0"/>
            <xsd:element name="SetChannelSubscriptionByName"
            type="typens:ArrayOfString" minOccurs="0"/>
          <xsd:element name="AddToGroupsById"
            type="typens:ArrayOfString"
            minOccurs="0"/>
          <xsd:element name="RemoveFromGroupsById"
            type="typens:ArrayOfString" minOccurs="0"/>
        <xsd:element name="SetGroupMembershipsById"
            type="typens:ArrayOfString" minOccurs="0"/>
          <xsd:element name="AssignRolesById"
            type="typens:ArrayOfString"
            minOccurs="0"/>
          <xsd:element name="DropRolesById"
            type="typens:ArrayOfString"
            minOccurs="0"/>
          <xsd:element name="SetRolesById"
            type="typens:ArrayOfString" minOccurs="0"/>
          <xsd:element name="SubscribeToChannelsById"
            type="typens:ArrayOfString" minOccurs="0"/>
          <xsd:element name="UnsubscribeFromChannelsbyId"
            type="typens:ArrayOfString" minOccurs="0"/>
          <xsd:element name="SetChannelSubscriptionById"
            type="typens:ArrayOfString" minOccurs="0"/>
          <xsd:element name="AddFileCreationPermissions"
            type="typens:ArrayOfPermission" minOccurs="0"/>
          <xsd:element name="RemoveFileCreationPermissions"
            type="typens:ArrayOfPermission" minOccurs="0"/>
          <xsd:element name="SetFileCreationPermissions"
            type="typens:ArrayOfPermission" minOccurs="0"/>
          <xsd:element name="SetPrinterOptions"
            type="typens:ArrayOfPrinterOptions" minOccurs="0"/>
          <xsd:element name="SetLicenseOptions"
            type="typens:ArrayOfString" minOccurs="0"/>
          <xsd:element name="AddLicenseOptions"
            type="typens:ArrayOfString" minOccurs="0"/>
          <xsd:element name="RemoveLicenseOptions"
            type="typens:ArrayOfString" minOccurs="0"/>
        </xsd:choice>
      </xsd:sequence>
    </xsd:complexType>
```

Request elements

SetAttributes
The general attributes to update.

AddToGroupsByName
The names of groups to which to add users. Specify either AddToGroupsByName or AddToGroupsById.

RemoveFromGroupsByName
The names of groups from which to remove users. Specify either RemoveFromGroupsByName or RemoveFromGroupsById.

SetGroupMembershipsByName
The groups for which to update users' membership. Specify either SetGroupMembershipsByName or SetGroupMembershipsById.

AssignRolesByName
The names of roles to assign to users. Specify either AssignRolesByName or AssignRolesById.

DropRolesByName
The names of roles from which to remove users. Specify either DropRolesByName or DropRolesById.

SetRolesByName
The names of roles to update. Specify either SetRolesByName or SetRolesById.

SubscribeToChannelsByName
The names of channels to which to subscribe users. Specify either SubscribeToChannelsByName or SubscribeToChannelsById.

UnsubscribeFromChannelsByName
The names of channels from which to unsubscribe users. Specify either UnsubscribeFromChannelsByName or UnsubscribeFromChannelsById.

SetChannelSubscriptionsByName
The names of channels to update for users. Specify either SetChannelSubscriptionsByName or SetChannelSubscriptionsById.

AddToGroupsById
The IDs of groups to which to add users. Specify either AddToGroupsById or AddToGroupsByName.

RemoveFromGroupsById
The IDs of groups from which to remove users. Specify either RemoveFromGroupsById or RemoveFromGroupsByName.

SetGroupMembershipsById
The IDs of groups for which to update users' membership. Specify either SetGroupMembershipsById or SetGroupMembershipsByName.

AssignRolesById
The IDs of roles to assign to users. Specify either AssignRolesById or AssignRolesByName.

DropRolesById
The IDs of roles from which to remove users. Specify either DropRolesById or DropRolesByName.

SetRolesById
The IDs of roles to update for users. Specify either SetRolesById or SetRolesByName.

SubscribeToChannelsById
The IDs of channels to which to subscribe users. Specify either SubscribeToChannelsById or SubscribeToChannelsByName.

UnsubscribeFromChannelsById
The IDs of channels from which to unsubscribe users. Specify either UnsubscribeFromChannelsById or UnsubscribeFromChannelsByName.

SetChannelSubscriptionsById
The IDs of channels to update for users. Specify either SetChannelSubscriptionsById or SetChannelSubscriptionsByName.

AddFileCreationPermissions
Grants users the permissions to add files.

RemoveFileCreationPermissions
Revokes users' ability to add files.

SetFileCreationPermissions
Modifies users' ability to add files.

SetPrinterOptions
The printer options to set for the users.

SetLicenseOptions
The license options to set for the users.

AddLicenseOptions
Grants users the right to add license options.

RemoveLicenseOptions
Removes the right of users to add license options.

UpdateUserOperationGroup

Specifies the UpdateUserOperation element to use during the UpdateUser operation.

Request schema

```
<xsd:complexType name="UpdateUserOperationGroup">
   <xsd:sequence>
      <xsd:element name="UpdateUserOperation"
         type="typens:UpdateUserOperation"
         maxOccurs="unbounded" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UpdateUserOperation
The UpdateUserOperation element for use with the UpdateUser operation.

UpdateVolumeProperties

Updates an Encyclopedia volume. To update a volume, specify the types of updates to make using UpdateVolumeOperationGroup, then specify which Encyclopedia volume to update.

Request schema

```
<xsd:complexType name="UpdateVolumeProperties">
   <xsd:sequence>
      <xsd:element name="UpdateVolumePropertiesOperationGroup"
         type="typens:UpdateVolumePropertiesOperationGroup"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UpdateVolumePropertiesOperationGroup
The tasks to perform.

UpdateVolumePropertiesOperation

Specifies the tasks to perform during the UpdateVolumeProperties operation.

Request schema

```
<xsd:complexType name="UpdateVolumePropertiesOperation">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="SetAttributes" type="typens:Volume"
            minOccurs="0"/>
         <xsd:element name="SetPrinterOptions"
            type="typens:ArrayOfPrinterOptions" minOccurs="0"/>
         <xsd:element name="SetSystemPrinters"
            type="typens:ArrayOfPrinter" minOccurs="0"/>
         <xsd:element name="ClearSystemPrinters"
            type="typens:ArrayOfString" minOccurs="0"/>
         <xsd:element name="SetAutoArchiveSchedules"
            type="typens:JobSchedule" minOccurs="0"/>
      </xsd:choice>
```

```
   </xsd:sequence>
</xsd:complexType>
```

Request elements

SetAttributes
Contains one or more of the following Encyclopedia volume properties to update:

- DefaultPrinterName
- MaxJobRetryCount
- JobRetryInterval
  The interval between retry attempts. Measured in seconds.
- DefaultViewingPreference
- DHTMLPageCaching
- DHTMLPageCachingExpiration
  If DHTMLPageCaching is True, set the DHTMLPageCachingExpiration to a valid value. To disable DHTMLPageCaching, set DHTMLPageCachingExpiration to -1.

SetPrinterOptions
The Encyclopedia volume printer options to update.

SetSystemPrinters
The printer to set as the system printer for the Encyclopedia volume.

ClearSystemPrinters
The printer to remove from the Encyclopedia volume.

SetAutoArchiveSchedules
The start of the autoarchive schedule for folders and files on the Encyclopedia volume.

UpdateVolumePropertiesOperationGroup

Specifies the UpdateVolumePropertiesOperation element to use during the UpdateVolumeProperties operation.

Request schema

```
<xsd:complexType name="UpdateVolumePropertiesOperationGroup">
   <xsd:sequence>
      <xsd:element name="UpdateVolumePropertiesOperation"
         type="typens:UpdateVolumePropertiesOperation"
         maxOccurs="unbounded" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UpdateVolumePropertiesOperation
The UpdateVolumePropertiesOperation element for use with the UpdateVolumeProperties operation.

UploadFile

Uploads a file to an Encyclopedia volume. You can upload the file as a MIME attachment or embed it in the request. To embed the file in the request, specify the ContentData element of the attachment.

Request schema

```
<xsd:complexType name="UploadFile">
   <xsd:sequence>
      <xsd:element name="NewFile" type="typens:NewFile"/>
      <xsd:element name="CopyFromLatestVersion"
         type="typens:ArrayOfString" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="Content" type="typens:Attachment"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

NewFile
The file to upload.

CopyFromLatestVersion
Copies one or more of the following properties from the latest version of the file, if one exists, to the version of the file that you upload:

- Description
  The description of the file
- Permissions
  Access control list (ACL) specifying the users and roles that can access the file
- ArchiveRule
  The autoarchive rules for the file

Content
The information about the file, such as the encoding the file uses and the data to upload.

Response schema

```
<xsd:complexType name="UploadFileResponse">
   <xsd:sequence>
      <xsd:element name="FileId" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

FileId
The ID of the uploaded file.

WaitForExecuteReport

Retrieves the status of the request to cancel synchronous report generation after receiving the Pending status. Send WaitForExecuteReport after sending CancelReport.

If progressive viewing is enabled, WaitForExecuteReport retrieves the status after the first page generates. Otherwise, WaitForExecuteReport waits until the report is complete.

If the current job status is Pending, WaitForExecuteReport waits for the report to generate.

Request schema

```
<xsd:complexType name="WaitForExecuteReport">
   <xsd:sequence>
      <xsd:element name="ObjectId" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ObjectId
The ID of the job.

Response schema

```
<xsd:complexType name="WaitForExecuteReportResponse">
   <xsd:sequence>
      <xsd:element name="Status"
         type="xsd:string"/>
      <xsd:element name="ErrorDescription" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="OutputFileType" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ObjectId" type="xsd:string"/>
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Status
The status of the request. One of the following values:

- Done
- Failed
- FirstPage

ErrorDescription
The description of the error. Returned if Status is Failed.

OutputFileType
The type of the output file.

ObjectId
The object ID of the report.

ConnectionHandle
The ID of the report. Supports viewing a report when the report is already in iHub System. Specified in the SOAP header.

Chapter

9

Actuate Information Delivery API data types

This chapter provides reference documentation for the data types the Actuate Information Delivery API uses. Each entry includes a general description of the data type, its schema, and a description of its elements.

AbsoluteDate

A complex data type that describes a date and run options for a job.

Schema

```
<xsd:complexType name="AbsoluteDate">
   <xsd:sequence>
      <xsd:element name="RunOn" type="xsd:string"/>
      <xsd:element name="OnceADay" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Repeat" type="typens:Repeat"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

RunOn
The date that a job is scheduled to run.

OnceADay
The days a job is to be run.

Repeat
Repeats the job run during a set start and stop time.

acDouble

A simple data type that represents a hexadecimal double.

Schema

```
<xsd:simpleType name="acDouble">
   <xsd:restriction base="xsd:hexBinary" />
</xsd:simpleType>
```

acNull

A simple data type that represents a null value.

Schema

```
<xsd:simpleType name="acNull">
   <xsd:restriction base="xsd:string">
      <xsd:maxLength value="0" />
   </xsd:restriction>
</xsd:simpleType>
```

Aggregation

A complex data type that describes the aggregation action to perform on a column.

Schema

```
<xsd:complexType name="Aggregation">
   <xsd:sequence>
      <xsd:element name="ColumnName" type="xsd:string"/>
      <xsd:element name="AggregationFunctions"
         type="typens:ArrayOfString">
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
```

Elements

ColumnName
The names of the columns on which to perform aggregation.

AggregationFunctions
The aggregation function to perform. Each column can have only one aggregation function. Valid functions are:

- MIN
- MAX
- AVG
- SUM

ArchiveRule

A complex data type that represents an archiving rule.

Schema

```
<xsd:complexType name="ArchiveRule">
   <xsd:sequence>
      <xsd:element name="FileType" type="xsd:string"/>
      <xsd:element name="NeverExpire" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="ExpireDependentFiles" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="ArchiveOnExpiration" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:choice minOccurs="0"/>
         <xsd:element name="ExpirationAge" type="xsd:long"
            minOccurs="0"/>
         <xsd:element name="ExpirationTime" type="xsd:dateTime"
            minOccurs="0"/>
      </xsd:choice>
```

```
      <xsd:element name="IsInherited" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="InheritedFrom" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

FileType
The file type. Cannot exceed 20 characters.

NeverExpire
Specifies whether the object expires.

ExpireDependentFiles
Specifies whether the object's dependent files expire when the object is expired.

ArchiveOnExpiration
Specifies whether the object is archived before it is expired.

ExpirationAge
The expiration age for the object.

ExpirationTime
The expiration time for the object.

IsInherited
Specifies whether the rule is inherited.

InheritedFrom
The object from which the rule is inherited.

Argument

A complex data type that represents an argument.

Schema

```
<xsd:complexType name="Argument">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="Value" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Name
The name of the argument.

Value
The value of the argument.

Arrays of data types

Data type definitions can be grouped into arrays. Each array has a specific definition for that data type.

The schema for an array of a data type generally follows the following pattern:

```
<xsd:complexType name="ArrayOfX">
   <xsd:sequence>
      <xsd:element name="X" type="typens:X"
         maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

In the above listing, X is the data type of object the array contains. For example, the XML for an array of Aggregation objects is:

```
<xsd:complexType name="ArrayOfAggregation">
   <xsd:sequence>
      <xsd:element name="Aggregation" type="typens:Aggregation"
         maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

The following data types have arrays defined in this manner:

- Aggregation
- ArchiveRule
- Argument
- Attachment
- Channel
- ChannelCondition
- ColumnDefinition
- ColumnSchema
- ComponentIdentifier
- DataExtractionFormat
- DataFilterCondition
- DataRow
- DataSortColumn
- DocumentConversionOptions
- FieldDefinition
- File
- JobProperties
- JobScheduleCondition
- JobScheduleDetail
- LicenseOption
- MDSInfo
- NameValuePair
- ParameterDefinition
- ParameterValue
- PendingSyncJob
- Permission
- Printer
- PrinterOptions
- PropertyValue
- Record
- ResourceGroup
- ResourceGroupSettings

- FileCondition
- FileContent
- FileType
- FilterCriteria
- Group
- GroupCondition
- Grouping
- JobCondition
- JobNotice
- JobNoticeCondition
- ResultSetSchema
- Role
- RoleCondition
- RunningJob
- ServerInformation
- ServerResourceGroupSetting
- Service
- SortColumn
- User
- UserCondition

Some array definitions are different from the ones listed above. These arrays have a type definition for the element other than what appears in the array name. For example, the ArrayOfDate is defined as:

```
<xsd:complexType name="ArrayOfDate">
   <xsd:sequence>
      <xsd:element name="Date" type="typens:string"
         maxOccurs="unbounded" minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

In this definition, the element name is Date, but its type is defined as a string. The ArrayOfDate type is defined as an array of string elements. The arrays in this format are listed in Table 9-1, along with the associated element type.

Table 9-1 Non-standard arrays

Array type	Element type
Date	string
String	string
Int	int
Component	ComponentType
Long	long

Attachment

A complex data type that describes the object in the attachment and contains the attachment as binary data.

Schema

```
<xsd:complexType name="Attachment">
   <xsd:all>
      <xsd:element name="ContentId" type="xsd:string"/>
      <xsd:element name="ContentType" type="xsd:string"/>
      <xsd:element name="ContentLength" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="ContentEncoding" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Locale" type="xsd:string" minOccurs="0"/>
      <xsd:element name="ContentData" type="xsd:base64Binary"
         minOccurs="0"/>
   </xsd:all>
</xsd:complexType>
```

Elements

ContentId
Maps to the attachment's MIME header. ContentId is required.

ContentType
The type of file to upload, such as binary.

ContentLength
The length of the object.

ContentEncoding
The encoding the object uses. Cannot exceed 10 characters.

Locale
The object locale.

ContentData
The attachment as binary data. Use ContentData to embed the file in the request.

BookMark

A complex data type that specifies the bookmark value, display name, and element type in a report design.

Schema

```
<xsd:complexType name="BookMark">
   <xsd:sequence>
      <xsd:element name="BookMarkValue" type="xsd:string"/>
      <xsd:element name="DisplayName" type="xsd:string"/>
      <xsd:element name="ElementType" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

BookMarkValue
The bookmark value, such as the page number.

DisplayName
The display name of the bookmark.

ElementType
The element or component type.

Boolean

A standard XML Boolean data type with a value of True or False.

CancelJobStatus

A simple data type that represents the status of a request to cancel a synchronous report.

Schema

```
<xsd:simpleType name="CancelJobStatus" base="xsd:string">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Succeeded"/>
      <xsd:enumeration value="Failed"/>
      <xsd:enumeration value="InActive"/>
   </xsd:restriction>
</xsd:simpleType>
```

Elements

Succeeded
The synchronous report generation was successfully cancelled.

Failed
The request to cancel a synchronous report failed.

InActive
The synchronous report generation is complete and cannot be cancelled.

Channel

A complex data type that describes a channel.

Schema

```
<xsd:complexType name="Channel">
   <xsd:all>
      <xsd:element name="Id" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Name" type="xsd:string" minOccurs="0"/>
      <xsd:element name="PollingInterval" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="Description" type="xsd:string"
         minOccurs="0"/>
```

```
      <xsd:element name="Expiration" type="xsd:long"minOccurs="0"/>
      <xsd:element name="SmallImageURL" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="LargeImageURL" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="UserPermissions" type="xsd:string"
         minOccurs="0"/>
   </xsd:all>
</xsd:complexType>
```

Elements

Id
The channel ID.

Name
The channel name. Cannot exceed 50 characters.

PollingInterval
The number of seconds that elapses until the next time the BIRT iHub refreshes the contents of the channel. The minimum value is 10 seconds.

Description
The description of the channel. Cannot exceed 500 characters.

Expiration
The number of seconds an item remains on the channel before the item is removed.

SmallImageURL
The URL of the small custom image for the channel. Cannot exceed 100 characters.

LargeImageURL
The URL of the large custom image for the channel. Cannot exceed 100 characters.

UserPermissions
The permissions the current user has on the channel.

ChannelCondition

A complex data type that represents fields on which a search can be performed and the condition to match.

Schema

```
<xsd:complexType name="ChannelCondition">
   <xsd:sequence>
      <xsd:element name="Field" type="typens:ChannelField">
      <xsd:element name="Match" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Field
The ChannelField that represents the field for the condition.

Match
The condition to match. If the condition includes special characters, for example a dash, either escape the special character with a backslash (\) or enclose it in brackets ([]). For example, to search for a file 05-25-04Report.rptdocument, specify one of the following:

```
05\ -25\ -04Report.rptdocument
05[-]25[-]04Report.rptdocument
```

ChannelField

A simple data type that represents the field of a ChannelCondition.

Schema

```
<xsd:simpleType name=”ChannelField”>
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Name"/>
      <xsd:enumeration value="PollingInterval"/>
      <xsd:enumeration value="Description"/>
      <xsd:enumeration value="Expiration"/>
      <xsd:enumeration value="SmallImageURL"/>
      <xsd:enumeration value="LargeImageURL"/>
   </xsd:restriction>
</xsd:simpleType>
```

ChannelSearch

A complex data type that represents a channel search.

Schema

```
<xsd:complexType name="ChannelSearch">
   <xsd:sequence>
      <xsd:choice minOccurs="0">
         <xsd:element name="Condition"
            type="typens:ChannelCondition"/>
         <xsd:element name="ConditionArray"
            type="typens:ArrayOfChannelCondition"/>
      </xsd:choice>
      <xsd:choice minOccurs="0">
         <xsd:element name="SubscribedUserId" type="xsd:string"/>
         <xsd:element name="SubscribedUserName" type="xsd:string"/>
         <xsd:element name="PrivilegeFilter"
            type="typens:PrivilegeFilter">
         </xsd:element>
```

```
      </xsd:choice>
      <xsd:element name="IncludeInheritedPrivilege"
         type="xsd:boolean" minOccurs="0"/>
      <xsd:element name="FetchSize" type="xsd:int" minOccurs="0"/>
      <xsd:element name="FetchDirection" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="CountLimit" type="xsd:int" minOccurs="0"/>
      <xsd:element name="FetchHandle" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Condition
The search condition.

ConditionArray
An array of search conditions.

SubscribedUserId
The ID of the user subscribed to the channel.

SubscribedUserName
The name of the user subscribed to the channel.

PrivilegeFilter
The privileges for which to search. Use PrivilegeFilter to determine the channels to which the specified user or role has the specified privileges.

IncludeInheritedPrivilege
Specifies whether to search only the privileges directly assigned to the channel or to include inherited privileges.

FetchSize
The maximum number of records to retrieve and return in a result set. The default value is 500.

FetchDirection
If True, records are retrieved forward. If False, records are retrieved backward. The default value is True.

CountLimit
The maximum number of records to count. By default, CountLimit is equal to FetchSize. To count all records, set CountLimit to -1.

FetchHandle
Retrieves more items from the result set. In the second and subsequent calls for data, specify the same search criteria as in the original call.

ColumnDefinition

A complex data type that describes a column in a query.

Schema

```
<xsd:complexType name="ColumnDefinition">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="Description" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="DisplayName" type="xsd:string"/>
      <xsd:element name="Heading" type="xsd:string" minOccurs="0"/>
      <xsd:element name="HelpText" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="DataType" type="typens:DataType"
         minOccurs="0"/>
      <xsd:element name="DisplayLength" type="xsd:long"
         minOccurs="0"/>
      </xsd:element>
      <xsd:element name="DisplayFormat" type="xsd:string"
         minOccurs="0"/>
   </xsd:element>
      <xsd:element name="AnalysisType" minOccurs="0">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="Automatic"/>
               <xsd:enumeration value="Dimension"/>
               <xsd:enumeration value="Measure"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="HorizontalAlignment" minOccurs="0">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="Automatic"/>
               <xsd:enumeration value="Left"/>
               <xsd:enumeration value="Center"/>
               <xsd:enumeration value="Right"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="EnableFilter" type="xsd:boolean"/>
      <xsd:element name="TextFormat" minOccurs="0">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="Plain" />
               <xsd:enumeration value="HTML" />
```

```
            <xsd:enumeration value="RTF" />
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:element>
   <xsd:element name="Wrap" minOccurs="0">
      <xsd:simpleType>
         <xsd:restriction base="xsd:string">
            <xsd:enumeration value="None" />
            <xsd:enumeration value="Word" />
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:element>
   <xsd:element name="CategoryPath" type="xsd:string"
      minOccurs="0" />
</xsd:sequence>
</xsd:complexType>
```

Elements

Name
The column name.

Description
The long description of the column.

DisplayName
The display name of the column. If not specified, the value of the Name element is used. If the query is performed on an information object (.iob) or data source map (.sma) file, DisplayName is used as the group label.

Heading
The text to display above the column in the output file.

HelpText
The text to display when the user holds the cursor over a column. For example, a value of a data column.

DataType
The data type of the column.

DisplayLength
The width of the column, in number of characters.

DisplayFormat
The format in which the data of the column appears in the output file.

AnalysisType
Specifies how data in the column is analyzed. One of the following values:

- Automatic
  Numeric values are analyzed as measures. Non-numeric values are analyzed as dimensions.

- Dimension
  Numeric values are analyzed as dimensions.
- Measure
  Numeric values are analyzed as measures.

HorizontalAlignment
Specifies how data in the column is aligned horizontally. One of the following values:

- Automatic
- Left
- Center
- Right

EnableFilter
Specifies whether filtering for the column is enabled. If True, the data in the column can be filtered.

TextFormat
Specifies text formatting. Valid values are:

- Plain
- HTML
- RTF

Wrap
Specifies word wrapping. Valid values are:

- None
- Word

CategoryPath
The category path for the column.

ColumnDetail

A complex data type that describes the type of data within a column.

Schema

```
<xsd:complexType name="ColumnDetail">
   <xsd:sequence>
      <xsd:element name="name" type="xsd:string" />
      <xsd:element name="type" type="typens:TypeName" />
      <xsd:element name="displayName" type="xsd:string" />
```

```
   </xsd:sequence>
</xsd:complexType>
```

Elements

name
The name of the column.

type
The type of data within the column.

displayName
The display name of the column.

ColumnSchema

A complex data type that describes the schema of a column.

Schema

```
<xsd:complexType name="ColumnSchema">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="Alias" type="xsd:string" minOccurs="0"/>
      <xsd:element name="DataType" type="xsd:int" minOccurs="0"/>
      <xsd:element name="TypeName" type="xsd:string"/>
      <xsd:element name="Label" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Visibility" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Name
The column name.

Alias
User-defined name for column.

DataType
The data type of the column.

TypeName
The name of the data type.

Label
The column label.

Visibility
Specifies whether column is visible. The default value is True.

ComponentIdentifier

A complex data type that identifies the component by ID or name.

Schema

```
<xsd:complexType name="ComponentIdentifier">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="Id" type="xsd:string"/>
         <xsd:element name="Name" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="DisplayName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ClassId" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Id
The ID of the component.

Name
The name of the component.

DisplayName
The display name of the component.

ClassId
The class ID of the component.

ComponentType

A complex data type that represents a component.

Schema

```
<xsd:complexType name="ComponentType">
   <xsd:all>
      <xsd:element name="Id" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Name" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Value" type="xsd:string" minOccurs="0"/>
      <xsd:element name="DisplayName" type="xsd:string"
         minOccurs="0" />
      <xsd:element name="ClassId" type="xsd:string"
         minOccurs="0"/>
   </xsd:all>
</xsd:complexType>
```

Elements

Id
The component ID.

Name
The component name.

Value
The component value.

DisplayName
The display name of the component.

ClassId
The class ID of the component.

CustomEvent

A complex data type that specifies information used within a custom event.

Schema

```
<xsd:complexType name="CustomEvent">
   <xsd:sequence>
      <xsd:element name="EventParameter" type="xsd:string" />
   </xsd:sequence>
</xsd:complexType>
```

Elements

EventParameter
A value used within the custom event.

Daily

A complex data type that describes daily types of job scheduling.

Schema

```
<xsd:complexType name="Daily">
   <xsd:sequence>
      <xsd:element name="FrequencyInDays" type="xsd:long"/>
      <xsd:element name="OnceADay" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Repeat" type="typens:Repeat"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

FrequencyInDays
The number of times a job is run daily in days.

OnceADay
A string representing when a job is to be run once a day.

Repeat
The number of times the schedule is repeated.

DatabaseConnectionDefinition

A complex data type that describes an Actuate Caching service (ACS) database connection object in the Encyclopedia volume.

Schema

```
<xsd:complexType name="DatabaseConnectionDefinition">
   <xsd:all>
      <xsd:element name="Name" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Id" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Type" type="xsd:string" minOccurs="0"/>
      <xsd:element name="ConfigKey" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ConnectionParameters"
         type="typens:ArrayOfParameterValue" minOccurs="0"/>
      <xsd:element name="DBUsername" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="DBPassword" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="DBAdminUsername" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="DBAdminPassword" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="DBLoadPath" type="xsd:string"
         minOccurs="0" />
   </xsd:all>
</xsd:complexType>
```

Elements

Name
The name of the data connection definition (.dcd) file to use for the connection.

Id
The ID of the DCD.

Type
The type of database to which to connect. The list of available database types is returned by GetDatabaseConnectionTypes.

ConfigKey
The ConfigKey for the database connection.

ConnectionParameters
Any parameters required to connect to the database.

DBUsername
The user name to use to access the database.

DBPassword
The password to use to access the database.

DBAdminUsername
The user name of the database administrator.

DBAdminPassword
The password of the database administrator.

DBLoadPath
The database load path.

DataCell

A complex data type describing the types of data within a data cell.

Schema

```
<xsd:complexType name="DataCell">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="int" type="xsd:int" />
         <xsd:element name="sht" type="xsd:short" />
         <xsd:element name="dbl" type="xsd:double" />
         <xsd:element name="dbn" type="typens:acDouble" />
         <xsd:element name="cur" type="xsd:string" />
         <xsd:element name="dtm" type="xsd:dateTime" />
         <xsd:element name="str" type="xsd:string" />
         <xsd:element name="bln" type="xsd:boolean" />
         <xsd:element name="nll" type="typens:acNull" />
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
```

DataExtractionFormat

A complex data type that describes the format of a file and its mime type.

Schema

```
<xsd:complexType name="DataExtractionFormat">
   <xsd:sequence>
      <xsd:element name="OutputFormat" type="xsd:string"/>
      <xsd:element name="MimeType" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

OutputFormat
The format of the file.

MimeType
The file mime type.

DataFilterCondition

A complex data type that describes a filter condition.

Schema

```
<xsd:complexType name="DataFilterCondition">
   <xsd:sequence>
      <xsd:element name="ColumnName" type="xsd:string"/>
      <xsd:element name="Operation" type="xsd:string"/>
      <xsd:element name="Operand1" type="xsd:string"/>
      <xsd:element name="Operand2" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Operand3" type="typens:ArrayOfString"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

ColumnName
The name of the column to filter.

Operation
The filtering operation.

Operand1
The first operand of the filter.

Operand2
The second operand of the filter.

Operand3
A list of operands for the filter.

DataRow

A complex data type that contains the information from a data row.

Schema

```
<xsd:complexType name="DataRow">
   <xsd:sequence>
      <xsd:element name="Cell" type="typens:DataCell"
         maxOccurs="unbounded" />
   </xsd:sequence>
</xsd:complexType>
```

Elements

Cell
A data cell from the row.

DataSchema

A complex data type that describes a data schema by column.

Schema

```
<xsd:complexType name="DataSchema">
   <xsd:sequence>
      <xsd:element name="Column" type="typens:ColumnDetail"
         maxOccurs="unbounded" minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Elements

Column
The schema of the information stored within a column.

DataSortColumn

A complex data type that describes a sorted data column.

Schema

```
<xsd:complexType name="DataSortColumn">
   <xsd:sequence>
      <xsd:element name="ColumnName" type="xsd:string"/>
      <xsd:element name="SortDirection">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="ASC"/>
               <xsd:enumeration value="DES"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
```

Elements

ColumnName
The name of the sorted column.

SortDirection
The direction of the sort. Valid values are:

- ASC - Ascending
- DES - Descending

DataSourceType

A simple data type that specifies the type of file in which a parameter exists.

Schema

```
<xsd:simpleType name="DataSourceType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="InfoObject"/>
      <xsd:enumeration value="ABInfoObject"/>
   </xsd:restriction>
</xsd:simpleType>
```

Elements

InfoObject
An information object.

ABInfoObject
An Actuate Basic information object.

DataType

A simple data type that specifies a data type.

Schema

```
<xsd:simpleType name="DataType"
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Currency"/>
      <xsd:enumeration value="Date"/>
      <xsd:enumeration value="DateOnly"/>
      <xsd:enumeration value="Time"/>
      <xsd:enumeration value="Double"/>
      <xsd:enumeration value="Integer"/>
      <xsd:enumeration value="String"/>
      <xsd:enumeration value="Boolean"/>
      <xsd:enumeration value="Structure"/>
      <xsd:enumeration value="Table"/>
   </xsd:restriction>
</xsd:simpleType>
```

Elements

Currency
A Currency data type.

Date
A Date data type.

DateOnly
A DateOnly data type.

Time
A Time data type.

Double
A Double data type.

Integer
An Integer data type.

String
A String data type.

Boolean
A Boolean data type.

Structure
A structure.

Table
A table.

DocumentConversionOptions

A complex data type that describes conversion options of a file.

Schema

```
<xsd:complexType name="DocumentConversionOptions">
   <xsd:sequence>
      <xsd:element name="FileType" type="xsd:string"/>
      <xsd:element name="OutputFormat" type="xsd:string"/>
      <xsd:element name="MimeType" type="xsd:string"/>
      <xsd:element name="Options"
         type="typens:ArrayOfParameterDefinition" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

FileType
The file type of the file.

OutputFormat
The output format of the file.

MimeType
The mime type of the file.

Options
The list of conversion options.

Event

A complex type that describes an event and its status.

Schema

```
<xsd:complexType name="Event">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="FileEvent" type="typens:FileEvent"
            minOccurs="0" />
```

```
            <xsd:element name="JobEvent" type="typens:JobEvent"
               minOccurs="0" />
            <xsd:element name="CustomEvent" type="typens:CustomEvent"
               minOccurs="0" />
         </xsd:choice>
         <xsd:element name="EventName" type="xsd:string" />
         <xsd:element name="EventType" type="typens:EventType" />
         <xsd:element name="PollingInterval" type="xsd:long"
            minOccurs="0" />
         <xsd:element name="PollingDuration" type="xsd:long"
            minOccurs="0" />
         <xsd:element name="LagTime" type="xsd:long" minOccurs="0" />
         <xsd:element name="EventStatus" minOccurs="0">
            <xsd:simpleType>
               <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="Uninitialized"/>
                  <xsd:enumeration value="Polling"/>
                  <xsd:enumeration value="Satisfied"/>
                  <xsd:enumeration value="Expired"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
      </xsd:sequence>
   </xsd:complexType>
```

Elements

FileEvent
Specifies information for a file event.

JobEvent
Specifies information for a job event.

CustomEvent
Specifies information for a custom event.

EventName
The name of the event.

EventType
The type of event.

PollingInterval
Specifies the amount of time to wait between event status checks. The minimum value is 10 seconds.

PollingDuration
Specifies the amount of time to check the event status.

LagTime
Specifies lag time value for the event.

EventStatus
The current status of the event. Valid values are:

- Uninitialized
- Polling
- Satisfied
- Expired

EventOptions

A complex data type that describes polling and other options for an event.

Schema

```
<xsd:complexType name="EventOptions">
   <xsd:sequence>
      <xsd:element name="DefaultEventPollingInterval"
         type="xsd:long" minOccurs="0" />
      <xsd:element name="DefaultEventPollingDuration"
         type="xsd:long" minOccurs="0" />
      <xsd:element name="DefaultEventLagTime" type="xsd:long"
         minOccurs="0" />
      <xsd:element name="EnableCustomEventService"
         type="xsd:boolean" minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Elements

DefaultEventPollingInterval
The amount of time to wait between polling the event.

DefaultEventPollingDuration
The duration of time to poll for an event.

DefaultEventLagTime
The amount of lag time for the event.

EnableCustomEventService
A flag indicating whether to enable the custom event service.

EventType

A simple data type that represents a type of event.

Schema

```
<xsd:simpleType name="EventType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="FileEvent" />
      <xsd:enumeration value="JobEvent" />
      <xsd:enumeration value="CustomEvent" />
      <xsd:enumeration value="NoEvent" />
   </xsd:restriction>
</xsd:simpleType>
```

Elements

FileEvent
A file type event.

JobEvent
A job type event.

CustomEvent
A custom type event.

NoEvent
No event.

ExecuteReportStatus

A simple data type that represents the status of report execution.

Schema

```
<xsd:simpleType name="ExecuteReportStatus">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Done"/>
      <xsd:enumeration value="Failed"/>
      <xsd:enumeration value="FirstPage"/>
      <xsd:enumeration value="Pending"/>
   </xsd:restriction>
</xsd:simpleType>
```

Elements

Done
The report execution succeeded.

Failed
The report execution failed.

FirstPage
The first page is complete. Applies only if progressive viewing is enabled.

Pending
The job is either in the queue or in the process of generating. Applies only if WaitTime is specified.

ExternalTranslatedRoleNames

A complex data type that represents one of the following roles:

- Administrator
- Operator
- All

Schema

```
<xsd:complexType name="ExternalTranslatedRoleNames">
   <xsd:sequence>
      <xsd:element name="Administrator" type="xsd:string"/>
      <xsd:element name="Operator" type="xsd:string"/>
      <xsd:element name="All" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

FieldDefinition

A complex data type that describes a scalar parameter.

Schema

```
<xsd:complexType name="FieldDefinition">
   <xsd:all>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="DisplayName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="DataType" type="typens:ScalarDataType"
         minOccurs="0"/>
      <xsd:element name="IsHidden" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="IsRequired" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="DefaultValue" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="SelectValueList"
         type="typens:ArrayOfString"
         minOccurs="0"/>
      <xsd:element name="FieldControlType" minOccurs="0">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="ControlList"/>
               <xsd:enumeration value="ControlListAllowNew"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
```

```
      <xsd:element name="SelectNameValueList"
         type="typens:ArrayOfNameValuePair" minOccurs="0" />
   </xsd:all>
</xsd:complexType>
```

Elements

Name
The name of the parameter.

DisplayName
The display name of the parameter.

DataType
The data type of the parameter. Valid values are:

- Currency
- Date
- Double
- Integer
- String
- Boolean

IsHidden
Specifies whether the parameter is hidden.

IsRequired
Specifies whether the parameter is required.

DefaultValue
The default value of the parameter.

SelectValueList
The list of available parameter values.

FieldControlType
The type of control used to represent the parameter. Valid values are:

- ControlListAllowNew
  A text box
- ControlList
  A drop-down list

SelectNameValueList
A list of name-value pairs used within the field.

FieldValue

A complex data type that describes a table parameter.

Schema

```
<xsd:complexType name="FieldValue">
   <xsd:all>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="Value" type="xsd:string"/>
   </xsd:all>
</xsd:complexType>
```

Elements

Name
The name of the parameter.

Value
The value of the parameter.

File

A complex data type that describes a file.

Schema

```
<xsd:complexType name="File">
   <xsd:all>
      <xsd:element name="Id" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Name" type="xsd:string" minOccurs="0"/>
      <xsd:element name="FileType" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Description" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="PageCount" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="Size" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="TimeStamp" type="xsd:dateTime"
         minOccurs="0"/>
      <xsd:element name="Version" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="VersionName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Owner" type="xsd:string" minOccurs="0"/>
      <xsd:element name="UserPermissions" type="xsd:string"
         minOccurs="0"/>
      </xsd:element>
```

```
      <xsd:element name="AccessType" type="typens:FileAccess"
        minOccurs="0"/>
   </xsd:all>
</xsd:complexType>
```

Elements

Id
The file ID.

Name
The name of the file. Actuate's internal data store imposes a fixed upper limit on the length of certain text strings.

FileType
The file type.

Description
The description of the file.

PageCount
The number of pages in the file.

Size
The size of the file.

TimeStamp
The time the file was created or modified, in Coordinated Universal Time (UTC).

Version
The version number.

VersionName
The version name.

Owner
The owner of the file.

UserPermissions
The current user's permissions for the file.

AccessType
The file's access type. Valid values are:

- Private
  Only the owner of the file and an administrator can access the file.
- Shared
  All users and roles specified in the access control list (ACL) for the file can access the file.

FileAccess

A simple data type that specifies the file's access type.

Schema

```
<xsd:simpleType name="FileAccess">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Private"/>
      <xsd:enumeration value="Shared"/>
   </xsd:restriction>
</xsd:simpleType>
```

Elements

Private
Only the owner of the file and an administrator can access the file.

Shared
All users and roles specified in the access control list (ACL) for the file can access the file.

FileCondition

A complex data type that represents the fields on which a search can be performed and the condition to match.

Schema

```
<xsd:complexType name="FileCondition">
   <xsd:sequence>
      <xsd:element name="Field" typens:FileField/>
      <xsd:element name="Match" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Field
File fields on which a search can be performed.

Match
The condition to match. If the condition includes special characters, for example a dash, either escape the special character with a backslash (\) or enclose it in brackets ([]). For example, to search for a file 05-25-04Report.rptdocument, specify one of the following:

```
05\ -25\ -04Report.rptdocument
05[-]25[-]04Report.rptdocument
```

FileContent

A complex data type that represents a list of attached files or the content of embedded files.

Schema

```
<xsd:complexType name="FileContent">
   <xsd:all>
      <xsd:element name="File" type="typens:File"/>
      <xsd:element name="Content" type="typens:Attachment"
         minOccurs="0"/>
   </xsd:all>
</xsd:complexType>
```

Elements

File
The attached file.

Content
The content of an embedded file.

FileEvent

A complex data type that contains information pertaining to file type events.

Schema

```
<xsd:complexType name="FileEvent">
   <xsd:sequence>
      <xsd:element name="MonitoredFilePath" type="xsd:string" />
   </xsd:sequence>
</xsd:complexType>
```

Elements

MonitoredFilePath
The file path of the file the event is monitoring.

FileField

A simple type that describes different fields that may exist for a file.

Schema

```
<xsd:simpleType name="FileField">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Name"/>
      <xsd:enumeration value="FileType"/>
      <xsd:enumeration value="Description"/>
      <xsd:enumeration value="PageCount"/>
      <xsd:enumeration value="Size"/>
      <xsd:enumeration value="TimeStamp"/>
```

```
      <xsd:enumeration value="Version"/>
      <xsd:enumeration value="VersionName"/>
      <xsd:enumeration value="Owner"/>
   </xsd:restriction>
</xsd:simpleType>
```

FileSearch

A complex data type that represents a file search.

Schema

```
<xsd:complexType name="FileSearch">
   <xsd:sequence>
      <xsd:choice minOccurs="0">
         <xsd:element name="Condition"
            type="typens:FileCondition">
         <xsd:element name="ConditionArray"
            type="typens:ArrayOfFileCondition"/>
      </xsd:choice>
      <xsd:element name="Owner" type="xsd:string" minOccurs="0"/>
      <xsd:choice minOccurs="0">
         <xsd:element name="DependentFileName" type="xsd:string"/>
         <xsd:element name="DependentFileId" type="xsd:string"/>
         <xsd:element name="RequiredFileName" type="xsd:string"/>
         <xsd:element name="RequiredFileId" type="xsd:string"/>
         <xsd:element name="PrivilegeFilter"
            type="typens:PrivilegeFilter"/>
      </xsd:choice>
      <xsd:element name="AccessType" type="typens:FileAccess"
         minOccurs="0"/>
      <xsd:element name="FetchSize" type="xsd:int" minOccurs="0"/>
      <xsd:element name="FetchDirection" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="CountLimit" type="xsd:int" minOccurs="0"/>
      <xsd:element name="FetchHandle" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="IncludeHiddenObject" type="xsd:boolean"
         minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Elements

Condition
The search condition.

ConditionArray
An array of search conditions. Use if search conditions apply to multiple fields.

Owner
The file owner.

DependentFileName
The name of the dependent file.

DependentFileId
The ID of the dependent file.

RequiredFileName
The name of the required file.

RequiredFileId
The ID of the required file.

PrivilegeFilter
The privileges for which to search. Use PrivilegeFilter to determine whether the specified user or role has the specified privileges on the file.

AccessType
The file's access type. Valid values are:

- Private
  Only the owner of the file and an administrator can access the file.
- Shared
  All users and roles specified in the access control list (ACL) for the file can access the file.

FetchSize
The maximum number of records to retrieve and return in a result set. The default value is 500.

FetchDirection
If True, records are retrieved forward. If False, records are retrieved backward. The default value is True.

CountLimit
The maximum number of records to count. By default, CountLimit is equal to FetchSize. To count all records, set CountLimit to -1.

FetchHandle
Retrieves more items from the result set. In the second and subsequent calls for data, specify the same search criteria as in the original call.

IncludeHiddenObject
Flag indicating if search should include hidden objects.

FileType

A complex data type that describes a file type. Specify FileType in the SOAP header for all execute, submit, and view IDAPI requests.

Schema

```
<xsd:complexType name="FileType">
   <xsd:all>
      <xsd:element name="Name" type="xsd:string" minOccurs="0"/>
      <xsd:element name="IsNative" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="IsExecutable" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="IsPrintable" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="IsRequired" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="OutputType" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="LocalExtension" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="DisplayType" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ShortDescription" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="LongDescription" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="SmallImageURL" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="LargeImageURL" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ExportBeforeViewing" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="DriverName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="MutexClass" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ContentType" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="EnableAutoParamCollection"
         type="xsd:boolean" minOccurs="0"/>
      <xsd:element name="IsCompoundDoc" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="AllowViewTimeParameter" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:all>
</xsd:complexType>
```

Elements

Name
The file type name.

IsNative
Specifies whether the file is an internal Actuate type. IsNative is read-only. Providing an input value for this attribute in CreateFileType or UpdateFileType causes SetAttributes to be ignored.

IsExecutable
Specifies whether the file is executable. If False, the file type is set to Document file type. The OutputType and ExportBeforeViewing attributes do not apply to Document file type.

IsPrintable
Specifies whether the file is printable. If the file type is Executable, IsPrintable refers to the output file.

IsRequired
Specifies whether the file is required.

OutputType
The file type for the output file. Required if the file type is Executable.

LocalExtension
The local extension.

DisplayType
Specifies either Simple or Advanced display types.

ShortDescription
The short description of the file type.

LongDescription
The long description of the file type.

SmallImageURL
The URL of the small image for the file.

LargeImageURL
The URL of the large image for the file.

ExportBeforeViewing
Specifies whether the file is exported before viewing.

DriverName
The name of the driver. Required if file type is Executable or Printable.

MutexClass
The mutex class name.

ContentType
The content type.

EnableAutoParamCollection
True enables automatic parameter collection for the file type.

IsCompoundDoc
Specifies whether the file is a compound document. The default value is False.

AllowViewTimeParameter
Specifies whether to allow view-time parameters. The default value is True.

FilterCriteria

A complex data type that describes the filter criteria for a query.

Schema

```
<xsd:complexType name="FilterCriteria">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="Value" type="xsd:string"/>
      <xsd:element name="Operation" type="xsd:string"/>
      <xsd:element name="PromptFilterCriteria" type="xsd:boolean"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Name
The name of the filter.

Value
The operand to use.

Operation
The operator to use. The available operators vary according to the data type of the column. Table 9-2 describes the available operators and the data types to which each operator applies.

Table 9-2 Filter criteria operators

Operator	Data types
=	String, Integer, Double, Currency, DateTime, Boolean
<	String, Integer, Double, Currency, DateTime, Boolean
<=	String, Integer, Double, Currency, DateTime, Boolean
>	String, Integer, Double, Currency, DateTime, Boolean
>=	String, Integer, Double, Currency, DateTime, Boolean

(continues)

Table 9-2 Filter criteria operators (continued)

Operator	Data types
<>	String, Integer, Double, Currency, DateTime
LIKE	String, Integer, Double, Currency, DateTime, Boolean
NOT LIKE	String
IN	String, Integer, Double, Currency, DateTime, Boolean
IS NULL	String, Integer, Double, Currency, DateTime, Boolean
IS NOT NULL	String, Integer, Double, Currency, DateTime, Boolean

PromptFilterCriteria
Specifies whether the user can change filter criteria. If True, the user can change the filter criteria.

FormatType

A simple data type that specifies a format type.

Schema

```
<xsd:simpleType name="FormatType">
   <xsd:restriction base="xsd:long">
      <xsd:enumeration value="0"/>
      <xsd:enumeration value="1"/>
      <xsd:enumeration value="2"/>
   </xsd:restriction>
</xsd:simpleType>
```

Elements

0
All formats.

1
View formats.

2
Search formats.

Group

A complex data type that describes a notification group.

Schema

```
<xsd:complexType name="Group">
   <xsd:sequence>
      <xsd:element name="Id" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Name" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Description" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Id
The group ID.

Name
The name of the group. Cannot exceed 50 characters.

Description
The description of the group. Cannot exceed 500 characters.

GroupCondition

A complex data type that represents the fields on which a search can be performed and the condition to match.

Schema

```
<xsd:complexType name="GroupCondition">
   <xsd:sequence>
      <xsd:element name="Field" type="typens:GroupField"/>
      <xsd:element name="Match" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Field
Fields on which a search can be performed.

Match
The condition to match. If the condition includes special characters, for example a dash, either escape the special character with a backslash (\) or enclose it in brackets ([]). For example, to search for a file 05-25-04Report.rptdocument, specify one of the following:

```
05\ -25\ -04Report.rptdocument
05[-]25[-]04Report.rptdocument
```

GroupField

A simple data type that represents fields on which a search can be performed.

Schema

```
<xsd:simpleType name="GroupField">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Name"/>
      <xsd:enumeration value="Description"/>
   </xsd:restriction>
</xsd:simpleType>
```

Grouping

A complex data type that describes how to group data in a query.

Schema

```
<xsd:complexType name="Grouping">
   <xsd:sequence>
      <xsd:element name="GroupKey" type="xsd:string"/>
      <xsd:element name="GroupSortOrder">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="ASC"/>
               <xsd:enumeration value="DES"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="GroupHeadingFields"
         type="typens:ArrayOfString" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

GroupKey
The key for grouping data.

GroupSortOrder
The grouping order. ASC specifies ascending order and DES specifies descending order. The values are case-sensitive. Specify in uppercase.

Differentiate sort order from fetch direction. Sort order specifies the order of the data returned by a query. Fetch direction specifies whether to get the next or previous set of results in a response when the result set exceeds the FetchSize.

GroupHeadingFields
The columns to include in the group.

GroupSearch

A complex data type that represents a notification group search.

Schema

```
<xsd:complexType name="GroupSearch">
   <xsd:sequence>
      <xsd:choice minOccurs="0">
         <xsd:element name="Condition"
            type="typens:GroupCondition"/>
         <xsd:element name="ConditionArray"
            type="typens:ArrayOfGroupCondition"/>
      </xsd:choice>
      <xsd:choice minOccurs="0">
         <xsd:element name="WithUserName" type="xsd:string"/>
         <xsd:element name="WithUserId" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="FetchSize" type="xsd:int" minOccurs="0"/>
      <xsd:element name="FetchDirection" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="CountLimit" type="xsd:int" minOccurs="0"/>
      <xsd:element name="FetchHandle" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Condition
The search condition.

ConditionArray
An array of search conditions.

WithUserName
The user name.

WithUserId
The user ID.

FetchSize
The maximum number of records to retrieve and return in a result set. The default value is 500.

FetchDirection
If True, records are retrieved forward. If False, records are retrieved backward. The default value is True.

CountLimit
The maximum number of records to count. By default, CountLimit is equal to FetchSize. To count all records, set CountLimit to -1.

FetchHandle
Retrieves more items from the result set. In the second and subsequent calls for data, specify the same search criteria as in the original call.

Header

The SOAP header that contains authentication data, locale information, and other required or optional data.

Schema

```
<xsd:complexType name="Header">
   <xsd:sequence>
      <xsd:element name="AuthId" type="xsd:string" />
      <xsd:element name="TargetVolume" type="xsd:string"
         minOccurs="0" />
      <xsd:element name="Locale" type="xsd:string" minOccurs="0" />
      <xsd:element name="ConnectionHandle" type="xsd:string"
         minOccurs="0" />
      <xsd:element name="TargetServer" type="xsd:string"
         minOccurs="0" />
      <xsd:element name="DelayFlush" type="xsd:boolean"
         minOccurs="0" />
      <xsd:element name="FileType" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="TargetResourceGroup" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="RequestID" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

AuthId
A system-generated, encrypted string. All requests except Login requests must have a valid AuthId in the SOAP header. The header passes this ID to BIRT iHub for validation.

TargetVolume
The Encyclopedia volume to which to direct the request.

Locale
Locale is used to format data using the language, date and time conventions, currency and other locale-specific conventions.

ConnectionHandle
An optional element that supports keeping a connection open to view a persistent report.

TargetServer
An optional element that refers to the BIRT iHub within a cluster to which to direct the request.

DelayFlush
A Boolean element that tells BIRT iHub to write updates to the disk when the value is False.

FileType
An optional element that supports specifying the file type to run, such as a BIRT design or HTML file.

TargetResourceGroup
An optional element that supports assigning a synchronous report generation request to a specific resource group at run time.

RequestID
A unique value that identifies the SOAP message.

Integer

A standard XML Integer data type that represents a number. Integer derives from Decimal data types by fixing the value of scale at 0.

InfoObjectData

A complex data type that describes the data from an information object.

Schema

```
<xsd:complexType name="InfoObjectData">
   <xsd:sequence>
      <xsd:element name="DataSchema" type="typens:DataSchema"
         minOccurs="0" />
      <xsd:element name="DataRows" type="typens:ArrayOfDataRow"
         minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Elements

DataSchema
The schema for the data rows.

DataRows
The data rows from the information object.

InfoObjectDataFormat

A simple data type that describes an information object's data format.

Schema

```
<xsd:simpleType name="InfoObjectDataFormat">
   <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="XML" />
      <xsd:enumeration value="CSV" />
   </xsd:restriction>
</xsd:simpleType>
```

Elements

XML
The file is in XML format.

CSV
The file is in comma separated values format.

JobCondition

A complex data type that represents the fields on which a search can be performed and the condition to match.

Schema

```
<xsd:complexType name="JobCondition">
   <xsd:sequence>
      <xsd:element name="Field" type="typens:JobField"/>
      <xsd:element name="Match" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Field
An element that includes one or more of the fields on which a search can be performed.

Match
The condition to match. If the condition includes special characters, for example a dash, either escape the special character with a backslash (\) or enclose it in brackets ([]). For example, to search for a file 05-25-04Report.rptdocument, specify one of the following:

```
05\ -25\ -04Report.rptdocument
05[-]25[-]04Report.rptdocument
```

JobEvent

A complex data type that represents the information pertaining to a job type event.

Schema

```
<xsd:complexType name="JobEvent">
   <xsd:sequence>
      <xsd:element name="JobId" type="xsd:string" />
```

```
      <xsd:element name="JobName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="JobStatus" type="typens:ArrayOfString"
         minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Elements

JobId
The ID of the job.

JobName
The job name.

JobStatus
The job status.

JobField

A simple data type that represents the fields on which a search can be performed.

Schema

```
<xsd:simpleType name="JobField">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="JobName"/>
      <xsd:enumeration value="Owner"/>
      <xsd:enumeration value="JobType"/>
      <xsd:enumeration value="Priority"/>
      <xsd:enumeration value="RoutedToNode"/>
      <xsd:enumeration value="StartTime"/>
      <xsd:enumeration value="DurationSeconds"/>
      <xsd:enumeration value="CompletionTime"/>
      <xsd:enumeration value="State"/>
      <xsd:enumeration value="NotifyCount"/>
      <xsd:enumeration value="OutputFileSize"/>
   </xsd:restriction>
</xsd:simpleType>
```

Elements

JobName
The job name.

Owner
The owner of the job.

JobType
The type of job. Valid values are:

- RunReport
- PrintReport

- RunAndPrintReport
- ConvertReport

Priority
The job priority.

RoutedToNode
The node to which the job is routed.

StartTime
The start time.

DurationSeconds
The job duration.

CompletionTime
The time the job is completed.

State
The state of the job. Valid values are:

- Scheduled
- Pending
- Running
- Succeeded
- Failed
- Cancelled
- Expired

NotifyCount
The number of notifications sent about the job.

OutputFileSize
The size of the output file.

JobInputDetail

A complex data type that describes the job input and output files.

Schema

```
<xsd:complexType name="JobInputDetail">
   <xsd:all>
      <xsd:element name="OutputFileVersionName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ReplaceLatestVersion" type="xsd:boolean"
         minOccurs="0"/>
```

```
<xsd:element name="OutputMaxVersion" type="xsd:int"
   minOccurs="0"/>
<xsd:element name="ValueFileType" minOccurs="0">
   <xsd:simpleType>
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="Temporary"/>
         <xsd:enumeration value="Permanent"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
<xsd:element name="ValueFileVersionName" type="xsd:string"
   minOccurs="0"/>
<xsd:element name="IsBundled" type="xsd:boolean"
   minOccurs="0"/>
<xsd:element name="RetryOption" type="typens:RetryOptionType"
   minOccurs="0"/>
<xsd:element name="MaxRetryCount" type="xsd:int"
   minOccurs="0"/>
<xsd:element name="RetryInterval" type="xsd:int"
   minOccurs="0"/>
<xsd:element name="MaxVersions" type="xsd:long"
   minOccurs="0"/>
<xsd:element name="NeverExpire" type="xsd:boolean"
   minOccurs="0"/
<xsd:element name="ArchiveRuleInherited" type="xsd:boolean"
   minOccurs="0"/>
<xsd:element name="ExpirationAge" type="xsd:int"
   minOccurs="0"/>
<xsd:element name="ExpirationDate" type="xsd:dateTime"
   minOccurs="0"/>
<xsd:element name="ArchiveOnExpire" type="xsd:boolean"
   minOccurs="0"/>
<xsd:element name="KeepWorkspace" type="xsd:boolean"
   minOccurs="0"/>
<xsd:element name="DriverTimeout" type="xsd:int"
   minOccurs="0"/>
<xsd:element name="PollingInterval" type="xsd:int"
   minOccurs="0"/>
<xsd:element name="DebugInstruction" type="xsd:string"
   minOccurs="0"/>
<xsd:element name="SendSuccessNotice" type="xsd:boolean"
   minOccurs="0"/>
<xsd:element name="SendFailureNotice" type="xsd:boolean"
   minOccurs="0"/>
<xsd:element name="SendEmailForSuccess" type="xsd:boolean"
   minOccurs="0"/>
<xsd:element name="SendEmailForFailure" type="xsd:boolean"
```

```
        minOccurs="0"/>
      <xsd:element name="AttachReportInEmail" type="xsd:boolean"
        minOccurs="0"/>
      <xsd:element name="OverrideRecipientPref" type="xsd:boolean"
        minOccurs="0"/>
      <xsd:element name="EmailFormat" type="xsd:string"
        minOccurs="0"/>
      <xsd:element name="RecordSuccessStatus" type="xsd:boolean"
        minOccurs="0"/>
      <xsd:element name="RecordFailureStatus" type="xsd:boolean"
        minOccurs="0"/>
      <xsd:element name="KeepOutputFile" type="xsd:boolean"
        minOccurs="0"/>
      <xsd:element name="ConversionOptions"
        type="typens:ConversionOptions" minOccurs="0"/>
      <xsd:element name="DataACL" type="typens:ArrayOfString"
        minOccurs="0"/>
    </xsd:all>
  </xsd:complexType>
```

Elements

OutputFileVersionName
The output file version.

ReplaceLatestVersion
Specifies whether to replace the latest version of the file with the current version.

OutputMaxVersion
The maximum number of versions to keep after a new version is generated.

ValueFileType
Specifies whether a value file is temporary or permanent.

ValueFileVersionName
The value file name.

IsBundled
Specifies whether the output object is bundled with the input object.

RetryOption
The retry settings. Valid values are:

- Disabled
- VolumeDefault
- Specified

MaxRetryCount
The maximum number of retry attempts.

RetryInterval
The interval between retry attempts. Measured in seconds.

MaxVersions
The maximum number of versions to keep in the Encyclopedia volume.

NeverExpire
Specifies whether the item expires.

ArchiveRuleInherited
Specifies whether the archive rules are inherited from another object.

ExpirationAge
Specifies the expiration age for the object.

ExpirationDate
The date when the job expires.

ArchiveOnExpire
Specifies whether the object is archived before it is expired.

KeepWorkspace
Specifies whether to keep or remove the workspace directory after executing the job.

DriverTimeout
The time for the driver to return from executing the job.

PollingInterval
The time interval to get status messages. The minimum value is 10 seconds.

DebugInstruction
The debug instructions.

SendSuccessNotice
Specifies whether success notices are sent if the job succeeds. Used only if OverrideRecipientPref is True.

SendFailureNotice
Specifies whether failure notices are sent if the job fails. Used only if OverrideRecipientPref is True.

SendEmailForSuccess
Specifies whether e-mail notifications are sent if the job succeeds. Used only if OverrideRecipientPref is True. If SendEmailForSuccess is True, e-mail notifications are sent to specified users and groups if the job succeeds. The default value is False.

SendEmailForFailure
Specifies whether e-mail notifications are sent if the job fails. Used only if OverrideRecipientPref is True. If SendEmailForFailure is True, e-mail

notifications are sent to specified users and groups if the job fails. The default value is False.

AttachReportInEmail
Specifies whether the output file is attached to the e-mail notification for successful jobs. Used only if OverrideRecipientPref is True. If AttachReportInEmail is True, the output file is attached to the e-mail notification. If False, only a link to the output file is sent. Specify the format for the attachment in the EmailFormat element. The default value is False.

OverrideRecipientPref
Specifies whether e-mail notifications and output attachments are sent according to job settings or user settings. If True, e-mail notifications and output attachments are sent according to job settings. If False, e-mail notifications and output attachments are sent according to user settings. The default value is False. If False, the following elements are ignored:

- AttachReportInEmail
- SendEmailForSuccess
- SendEmailForFailure
- SendSuccessNotice
- SendFailureNotice

EmailFormat
Specifies the output format of the report attached to the e-mail notification. Valid formats are:

- Excel (XLS)
- Excel (XLSX)
- PDF
- PostScript (PS)
- Word (DOC)
- Word (DOCX)
- PowerPoint (PPT)
- PowerPoint (PPTX)

RecordSuccessStatus
Specifies whether to record job success notices.

RecordFailureStatus
Specifies whether to record job failure notices.

KeepOutputFile
Specifies whether the generated output file remains in the Encyclopedia volume if the generation request succeeds but the printing request fails. Used if the job is to be generated and printed. If True, the output file remains in the Encyclopedia volume. If False, the output file is deleted if the printing request fails. The default value is False.

ConversionOptions
Specifies options for converting a report object instance (.roi) output to another format.

DataACL
Specifies the access control list (ACL) restricting data privileges.

JobNotice

A complex data type that describes a job notice.

Schema

```
<xsd:complexType name="JobNotice">
   <xsd:all>
      <xsd:element name="JobId" type="xsd:string" minOccurs="0"/>
      <xsd:element name="JobName" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Headline" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="JobState" minOccurs="0">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="Succeeded"/>
               <xsd:enumeration value="Failed"/>
               <xsd:enumeration value="Cancelled"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="CompletionTime" type="xsd:dateTime"
         minOccurs="0"/>
      <xsd:element name="ActualOutputFileName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="OutputFileName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="OutputFileVersionName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ActualOutputFileSize"
         type="xsd:unsignedLong"minOccurs="0"/>
      <xsd:element name="ActualOutputFileId" type="xsd:string"
         minOccurs="0"/>
```

```
      <xsd:element name="NotifiedUserId" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="NotifiedUserName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="NotifiedChannelId" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="NotifiedChannelName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="OutputFileVersion" type="xsd:long"
         minOccurs="0"/>
   </xsd:all>
</xsd:complexType>
```

Elements

JobId
The job ID.

JobName
The name of the job.

Headline
The job headline.

JobState
The state of the job. Valid values are:

- Succeeded
- Failed
- Cancelled

CompletionTime
The time the job is completed.

ActualOutputFileName
The output file name that the BIRT iHub generated.

OutputFileName
The output file name.

OutputFileVersionName
The output file version name.

ActualOutputFileSize
The size of the output file.

ActualOutputFileId
The output file ID that the BIRT iHub generated.

NotifiedUserId
The ID of the user who received the notice.

NotifiedUserName
The name of the user who received the notice.

NotifiedChannelId
The ID of the channel that received the notice.

NotifiedChannelName
The name of the channel that received the notice.

OutputFileVersion
The output file version number.

JobNoticeCondition

A complex data type that represents the fields on which a search can be performed and the condition to match.

Schema

```
<xsd:complexType name="JobNoticeCondition">
   <xsd:sequence>
      <xsd:element name="Field">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="JobId"/>
               <xsd:enumeration value="JobName"/>
               <xsd:enumeration value="OutputFileName"/>
               <xsd:enumeration value="JobState"/>
               <xsd:enumeration value="HeadLine"/>
               <xsd:enumeration value="CompletionTime"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="Match" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Field
The fields on which a search can be performed.

Match
The condition to match. If the condition includes special characters, for example a dash, either escape the special character with a backslash (\) or enclose it in brackets ([]). For example, to search for a file 05-25-04Report.rptdocument, specify one of the following:

```
05\ -25\ -04Report.rptdocument
05[-]25[-]04Report.rptdocument
```

JobNoticeField

A simple data type that represents the fields on which a search can be performed.

Schema

```
<xsd:simpleType name="JobNoticeField">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="JobId"/>
      <xsd:enumeration value="JobName"/>
      <xsd:enumeration value="OutputFileName"/>
      <xsd:enumeration value="JobState"/>
      <xsd:enumeration value="HeadLine"/>
      <xsd:enumeration value="CompletionTime"/>
   </xsd:restriction>
</xsd:simpleType>
```

JobNoticeSearch

A complex data type that represents the job notice search.

Schema

```
<xsd:complexType name="JobNoticeSearch">
   <xsd:sequence>
      <xsd:choice minOccurs="0">
         <xsd:element name="Condition"
            type="typens:JobNoticeCondition"/>
         <xsd:element name="ConditionArray"
            type="typens:ArrayOfJobNoticeCondition"/>
      </xsd:choice>
      <xsd:choice minOccurs="0">
         <xsd:element name="NotifiedUserId" type="xsd:string"/>
         <xsd:element name="NotifiedUserName" type="xsd:string"/>
         <xsd:element name="NotifiedChannelId" type="xsd:string"/>
         <xsd:element name="NotifiedChannelName"
            type="xsd:string">
      </xsd:choice>
      <xsd:element name="FetchSize" type="xsd:int" minOccurs="0"/>
      <xsd:element name="FetchDirection" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="CountLimit" type="xsd:int" minOccurs="0"/>
      <xsd:element name="FetchHandle" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Condition
The search condition.

ConditionArray
An array of search conditions.

NotifiedUserId
The ID of the user who received the notice.

NotifiedUserName
The name of the user who received the notice.

NotifiedChannelId
The ID of the channel that received the notice.

NotifiedChannelName
The name of the channel that received the notice.

FetchSize
The maximum number of records to retrieve and return in a result set. The default value is 500.

FetchDirection
If True, records are retrieved forward. If False, records are retrieved backward. The default value is True.

CountLimit
The maximum number of records to count. By default, CountLimit is equal to FetchSize. To count all records, set CountLimit to -1.

FetchHandle
Retrieves more items from the result set. In the second and subsequent calls for data, specify the same search criteria as in the original call.

JobPrinterOptions

A complex data type that describes the job printer options.

Schema

```
<xsd:complexType name="JobPrinterOptions">
   <xsd:sequence>
      <xsd:element name="PrinterName" type="xsd:string"/>
      <xsd:element name="Orientation" type="xsd:string"
         inOccurs="0"/>
      <xsd:element name="PageSize" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Scale" type="xsd:long" minOccurs="0"/>
      <xsd:element name="Resolution" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="NumberOfCopies" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="CollationOption" type="xsd:boolean"
```

```
            minOccurs="0"/>
         <xsd:element name="PaperTray" type="xsd:boolean"
            minOccurs="0"/>
         <xsd:element name="Duplex" type="xsd:string" minOccurs="0"/>
         <xsd:element name="IsColor" type="xsd:boolean"
            minOccurs="0"/>
         <xsd:element name="PaperLength" type="xsd:long"
            minOccurs="0"/>
         <xsd:element name="PaperWidth" type="xsd:long"
            minOccurs="0"/>
         <xsd:element name="PageRange" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="FormName" type="xsd:string"
            minOccurs="0"/>
         <xsd:element name="PrintToFile" type="xsd:string"
            minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
```

Elements

PrinterName
The printer name.

Orientation
The paper orientation.

PageSize
The page size.

Scale
The scaling factor.

Resolution
The resolution.

NumberOfCopies
The number of copies.

CollationOption
Specifies whether the printer's collation property is set.

PaperTray
Specifies whether the printer's paper tray option is set.

Duplex
The value of the printer's duplex property.

IsColor
Specifies whether the printer can print in color.

PaperLength
The paper length.

PaperWidth
The paper width.

PageRange
The page range. Cannot exceed 20 characters.

FormName
The form name.

PrintToFile
The setting of the print to file property. Cannot exceed 256 characters.

JobProperties

A complex data type that specifies the general job attributes, such as input document file name, output file name, and job execution status.

Schema

```
<xsd:complexType name="JobProperties">
   <xsd:sequence>
      <xsd:element name="JobId" type="xsd:string" minOccurs="0"/>
      <xsd:element name="JobName" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Priority" type="xsd:long" minOccurs="0"/>
      <xsd:element name="ResourceGroup" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Owner" type="xsd:string" minOccurs="0"/>
      <xsd:element name="JobType" minOccurs="0">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="RunReport"/>
               <xsd:enumeration value="PrintReport"/>
               <xsd:enumeration value="RunAndPrintReport"/>
               <xsd:enumeration value="ConvertReport"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="State" minOccurs="0">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="Scheduled"/>
               <xsd:enumeration value="Pending"/>
               <xsd:enumeration value=”Waiting”/>
               <xsd:enumeration value="Running"/>
               <xsd:enumeration value="Succeeded"/>
               <xsd:enumeration value="Failed"/>
```

```
            <xsd:enumeration value="Cancelled"/>
            <xsd:enumeration value="Expired"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="InputFileId" type="xsd:string"
    minOccurs="0"/>
<xsd:element name="InputFileName" type="xsd:string"
    minOccurs="0"/>
<xsd:element name="RunLatestVersion" type="xsd:boolean"
    minOccurs="0"/>
<xsd:element name="ParameterFileId" type="xsd:string"
    minOccurs="0"/>
<xsd:element name="ParameterFileName" type="xsd:string"
    minOccurs="0"/>
<xsd:element name="ActualOutputFileId"type="xsd:string"
    minOccurs="0"/>
<xsd:element name="ActualOutputFileName" type="xsd:string"
    minOccurs="0"/>
<xsd:element name="RequestedOutputFileName" type="xsd:string"
    minOccurs="0"/>
<xsd:element name="OutputFileVersionName" type="xsd:string"
    minOccurs="0"/>
<xsd:element name="SubmissionTime" type="xsd:dateTime"
    minOccurs="0"/>
<xsd:element name="CompletionTime" type="xsd:dateTime"
    minOccurs="0"/>
<xsd:element name="PageCount" type="xsd:long" minOccurs="0"/>
<xsd:element name="OutputFileSize" type="xsd:long"
    minOccurs="0"/>
<xsd:element name="RoutedToNode" type="xsd:string"
    minOccurs="0"/>
<xsd:element name="DurationSeconds" type="xsd:long"
    minOccurs="0"/>
<xsd:element name="StartTime" type="xsd:dateTime"
    minOccurs="0"/>
<xsd:element name="NextStartTime" type="xsd:dateTime"
    minOccurs="0"/>
<xsd:element name="RequestedHeadline" type="xsd:string"
    minOccurs="0"/>
<xsd:element name="ActualHeadline" type="xsd:string"
    minOccurs="0"/>
<xsd:element name="NotifyCount" type="xsd:string"
    minOccurs="0"/>
<xsd:element name="EventName" type="xsd:string"
    minOccurs="0"/>
<xsd:element name="EventType" type="typens:EventType"
```

```
        minOccurs="0"/>
        minOccurs="0"/>
      <xsd:element name="EventStatus" type="xsd:string"
        minOccurs="0"/>
      <xsd:element name="EventParameter" type="xsd:string"
        minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

JobId
The job ID.

JobName
The name of the job.

Priority
The job priority.

ResourceGroup
The name of the resource group to which a job is assigned, if any.

Owner
The owner of the job.

JobType
The type of job. Valid values are:

- RunReport
- PrintReport
- RunAndPrintReport
- ConvertReport

State
The state of the job. Valid values are:

- Scheduled
- Pending
- Running
- Succeeded
- Failed
- Cancelled
- Expired

InputFileId
The input file ID.

InputFileName
The input file full name and version number.

RunLatestVersion
Specifies whether to run the latest version of the file.

ParameterFileId
The parameter file ID. Exists only if the parameter file is specified.

ParameterFileName
The parameter file name. Exists only if the parameter file is specified.

ActualOutputFileId
The output file ID that the BIRT iHub generated.

ActualOutputFileName
The output file name that the BIRT iHub generated. This might be different than the RequestedOutputFileName.

RequestedOutputFileName
The requested name for the output file.

OutputFileVersionName
The output file version name.

SubmissionTime
The time the job was submitted.

CompletionTime
The time the job is completed.

PageCount
The number of pages.

OutputFileSize
The size of the output file.

RoutedToNode
The node to which the job is routed.

DurationSeconds
The job duration.

StartTime
The start time.

NextStartTime
The next time the job is scheduled to run. Applies only to scheduled jobs.

RequestedHeadline
The headline for the job.

ActualHeadline
The headline that the BIRT iHub generated.

NotifyCount
The number of notifications sent about the job.

EventName
The name of the job event.

EventType
The job event type.

EventStatus
The job event status.

EventParameter
The parameter for the job event.

JobSchedule

A complex data type that represents details about a job schedule.

Schema

```
<xsd:complexType name="JobSchedule">
   <xsd:sequence>
      <xsd:element name="TimeZoneName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ScheduleDetails"
         type="typens:ArrayOfJobScheduleDetail"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

TimeZoneName
The time zone. Cannot exceed 32 characters.

ScheduleDetails
The schedule details.

JobScheduleCondition

A complex data type that represents the fields on which a search can be performed and the condition to match.

Schema

```
<xsd:complexType name="JobScheduleCondition">
   <xsd:sequence>
      <xsd:element name="Field" type="typens:JobScheduleField"/>
      <xsd:element name="Match" type="xsd:string"/>
   </xsd:sequence>
```

```
   <xsd:complexType name="ArrayOfJobScheduleCondition">
      <xsd:sequence>
         <xsd:element name="JobScheduleCondition"
            type="typens:JobScheduleCondition" maxOccurs="unbounded"
            minOccurs="0"/>
      </xsd:sequence>
</xsd:complexType>
```

Elements

Field
Fields on which a search can be performed.

Match
The condition to match. If the condition includes special characters, for example a dash, either escape the special character with a backslash (\) or enclose it in brackets ([]). For example, to search for a file 05-25-04Report.rptdocument, specify one of the following:

```
05\ -25\ -04Report.rptdocument
05[-]25[-]04Report.rptdocument
```

ArrayOfJobScheduleCondition
Fields on which a job schedule search can be performed.

JobScheduleDetail

A complex data type that specifies a schedule for running a job.

Schema

```
<xsd:complexType name="JobScheduleDetail">
   <xsd:sequence>
      <xsd:element name="ScheduleType">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="AbsoluteDate"/>
               <xsd:enumeration value="Daily"/>
               <xsd:enumeration value="Weekly"/>
               <xsd:enumeration value="Monthly"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="ScheduleStartDate" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ScheduleEndDate" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="DatesExcluded" type="typens:ArrayOfDate"
         minOccurs="0"/>
      <xsd:choice>
         <xsd:element name="AbsoluteDate"
```

```
            type="typens:AbsoluteDate"/>
         <xsd:element name="Daily" type="typens:Daily"/>
         <xsd:element name="Weekly" type="typens:Weekly"/>
         <xsd:element name="Monthly" type="typens:Monthly"/>
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
```

Elements

ScheduleType
The type of schedule. Valid values are:

- AbsoluteDate
- Daily
- Weekly
- Monthly

ScheduleStartDate
The date on which to start the schedule. Express the date as a standard XML String data type in the format YYYY-MM-DDThh:mm:ss-sss. Milliseconds, if specified, are ignored.

ScheduleEndDate
The date on which to end the schedule. Express the date as a standard XML String data type using the format YYYY-MM-DDThh:mm:ss-sss. Milliseconds, if specified, are ignored.

DatesExcluded
An array of dates to exclude from the schedule.

JobScheduleField

A simple data type describing fields upon which a search can be performed.

Schema

```
<xsd:simpleType name="JobScheduleField">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="JobName"/>
      <xsd:enumeration value="Owner"/>
      <xsd:enumeration value="JobType"/>
      <xsd:enumeration value="Priority"/>
      <xsd:enumeration value="NextStartTime"/>
      <xsd:enumeration value="State"/>
      <xsd:enumeration value="ParameterFileId"/>
   </xsd:restriction>
</xsd:simpleType>
```

Elements

JobName
The name of the job.

Owner
The owner of the job.

JobType
The type of job. Valid values are:

- RunReport
- PrintReport
- RunAndPrintReport
- ConvertReport

Priority
The job priority.

NextStartTime
The next time the job is scheduled to run. Applies only to scheduled jobs.

State
The state of the job. Valid values are:

- Scheduled
- Pending
- Running
- Succeeded
- Failed
- Cancelled
- Expired

ParameterFileId
The parameter file ID. Exists only if the parameter file is specified.

JobScheduleSearch

A complex data type that represents a job schedule search.

Schema

```
<xsd:complexType name="JobScheduleSearch">
   <xsd:sequence>
      <xsd:choice minOccurs="0">
         <xsd:element name="Condition"
            type="typens:JobScheduleCondition"/>
         <xsd:element name="ConditionArray"
            type="typens:ArrayOfJobScheduleCondition"/>
      </xsd:choice>
```

```
        <xsd:element name="RequestedOutputFileName" type="xsd:string"
           minOccurs="0"/>
        <xsd:element name="InputFileName" type="xsd:string"
           minOccurs="0"/>
        <xsd:element name="InputFileId" type="xsd:string"
           minOccurs="0"/>
        <xsd:element name="EventType" type="typens:EventType"
           minOccurs="0" />
        <xsd:choice minOccurs="0">
           <xsd:element name="NotifiedUserId" type="xsd:string"/>
           <xsd:element name="NotifiedUserName" type="xsd:string"/>
           <xsd:element name="NotifiedChannelId"
              type="xsd:string">
           <xsd:element name="NotifiedChannelName"
              type="xsd:string"/>
        </xsd:choice>
        <xsd:element name="FetchSize" type="xsd:int" minOccurs="0"/>
        <xsd:element name="FetchDirection" type="xsd:boolean"
           minOccurs="0"/>
        <xsd:element name="CountLimit" type="xsd:int" minOccurs="0"/>
        <xsd:element name="FetchHandle" type="xsd:string"
           minOccurs="0"/>
     </xsd:sequence>
  </xsd:complexType>
```

Elements

Condition
The search condition.

ConditionArray
The array of search conditions.

RequestedOutputFileName
The output file name.

InputFileName
The input file name.

InputFileId
The input file ID.

EventType
The event type of the job.

NotifiedUserId
The ID of the user to notify.

NotifiedUserName
The name of the user to notify.

NotifiedChannelId
The ID of the channel to notify.

NotifiedChannelName
The name of the channel to notify.

FetchSize
The maximum number of records to retrieve and return in a result set. The default value is 500.

FetchDirection
If True, records are retrieved forward. If False, records are retrieved backward. The default value is True.

CountLimit
The maximum number of records to count. By default, CountLimit is equal to FetchSize. To count all records, set CountLimit to -1.

FetchHandle
Retrieves more items from the result set. In the second and subsequent calls for data, specify the same search criteria as in the original call.

JobSearch

A complex data type that represents a job search.

Schema

```
<xsd:complexType name="JobSearch">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="Condition" type="typens:JobCondition"/>
         <xsd:element name="ConditionArray"
            type="typens:ArrayOfJobCondition"/>
      </xsd:choice>
      <xsd:element name="Owner" type="xsd:string" minOccurs="0" />
      <xsd:element name="ActualOutputFileName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ActualOutputFileId" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="RequestedOutputFileName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="InputFileName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="InputFileId" type="xsd:string"
         minOccurs="0"/>
      <xsd:choice minOccurs="0">
         <xsd:element name="NotifiedUserId" type="xsd:string"/>
         <xsd:element name="NotifiedUserName" type="xsd:string"/>
```

```
            <xsd:element name="NotifiedChannelId"
               type="xsd:string">
            <xsd:element name="NotifiedChannelName"
               type="xsd:string"/>
         </xsd:choice>
         <xsd:element name="FetchSize" type="xsd:int"
            minOccurs="0"/>
         <xsd:element name="FetchDirection" type="xsd:boolean"
            minOccurs="0"/>
         <xsd:element name="CountLimit" type="xsd:int"
            minOccurs="0"/>
         <xsd:element name="FetchHandle" type="xsd:string"
            minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
```

Elements

Condition
The search condition.

ConditionArray
The array of search conditions.

Owner
The owner of the job.

ActualOutputFileName
The output file name that the BIRT iHub generated.

ActualOutputFileId
The output file ID that the BIRT iHub generated.

RequestedOutputFileName
The output file requested name.

InputFileName
The input file name.

InputFileId
The input file ID.

NotifiedUserId
The ID of the user who received notification.

NotifiedUserName
The name of the user who received notification.

NotifiedChannelId
The ID of the channel that received notification.

NotifiedChannelName
The name of the channel that received notification.

FetchSize
The maximum number of records to retrieve and return in a result set. The default value is 500.

FetchDirection
If True, records are retrieved forward. If False, records are retrieved backward. The default value is True.

CountLimit
The maximum number of records to count. By default, CountLimit is equal to FetchSize. To count all records, set CountLimit to -1.

FetchHandle
Retrieves more items from the result set. In the second and subsequent calls for data, specify the same search criteria as in the original call.

LicenseOption

A complex data type that represents a license option.

Schema

```
<xsd:complexType name="LicenseOption">
   <xsd:all>
      <xsd:element name="Name" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Description" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ShortDescription" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Value" type="xsd:string" minOccurs="0"/>
   </xsd:all>
</xsd:complexType>
```

Elements

Name
The name of the license option.

Description
The description of the option.

ShortDescription
A shorter description of the option.

Value
The value of the option.

MDSInfo

A complex data type that describes a Message Distribution service (MDS).

Schema

```
<xsd:complexType name="MDSInfo">
   <xsd:sequence>
      <xsd:element name="ServerName" type="xsd:string"/>
      <xsd:element name="MDSIPAddress" type="xsd:string"/>
      <xsd:element name="MDSPortNumber" type="xsd:int"/>
      <xsd:element name="ServerState" type="typens:ServerState"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

ServerName
The server name.

MDSIPAddress
The IP address of the MDS.

MDSPortNumber
The port number the MDS uses.

ServerState
The server state.

Monthly

A complex data type that describes monthly job scheduling.

Schema

```
<xsd:complexType name="Monthly">
   <xsd:sequence>
      <xsd:element name="FrequencyInMonths" type="xsd:long" />
      <xsd:element name="OnDay" minOccurs="0">
         <xsd:simpleType>
            <xsd:restriction base="xsd:int">
               <xsd:minInclusive value="0" />
               <xsd:maxInclusive value="31" />
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="OnWeekDay" minOccurs="0">
         <xsd:simpleType>
            <xsd:restriction base="xsd:int">
               <xsd:minInclusive value="0" />
               <xsd:maxInclusive value="23" />
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="RunOn" minOccurs="0">
         <xsd:complexType>
            <xsd:sequence>
```

```
                <xsd:element name="WeekDay">
                   <xsd:simpleType>
                      <xsd:restriction base="xsd:string">
                         <xsd:enumeration value="Mon" />
                         <xsd:enumeration value="Tue" />
                         <xsd:enumeration value="Wed" />
                         <xsd:enumeration value="Thu" />
                         <xsd:enumeration value="Fri" />
                         <xsd:enumeration value="Sat" />
                         <xsd:enumeration value="Sun" />
                      </xsd:restriction>
                   </xsd:simpleType>
                </xsd:element>
                <xsd:element name="Occurrence">
                   <xsd:simpleType>
                      <xsd:restriction base="xsd:string">
                         <xsd:enumeration value="First" />
                         <xsd:enumeration value="Second" /> )
                         <xsd:enumeration value="Third" />
                         <xsd:enumeration value="Fourth" />
                         <xsd:enumeration value="Last" />
                      </xsd:restriction>
                   </xsd:simpleType>
                </xsd:element>
             </xsd:sequence>
          </xsd:complexType>
       </xsd:element>
       <xsd:element name="OnceADay" type="xsd:string"
          minOccurs="0" />
       <xsd:element name="Repeat" type="typens:Repeat"
          minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

FrequencyInMonths
The amount of times a job is to be run, in months.

OnDay
The day of the month the job is to run.

RunOn
Specifies what day of a week to run a job on, and which day of the month to run it. For example, the third Tuesday of the month.

OnceADay
Specifies the time the job is to be run.

Repeat
Specifies how often the schedule is to be repeated.

NameValuePair

A complex data type that represents a named piece of data and its value.

Schema

```
<xsd:complexType name="NameValuePair">
   <xsd:all>
      <xsd:element name="Name" type="xsd:string" />
      <xsd:element name="Value" type="xsd:string" nillable="true"/>
   </xsd:all>
  </xsd:complexType>
```

Elements

Name
The data's name.

Value
The data's value.

NewFile

A complex data type that describes a file.

Schema

```
<xsd:complexType name="NewFile">
   <xsd:all>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="VersionName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ReplaceExisting" type="xsd:Boolean"
         minOccurs="0"/>
      <xsd:element name="Versioning" type="typens:VersioningOption"
         minOccurs="0"/>
      <xsd:element name="MaxVersions"type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="Description" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ArchiveRule" type="typens:ArchiveRule"
         minOccurs="0"/>
      <xsd:element name="ACL" type="typens:ArrayOfPermission"
         minOccurs="0"/>
      <xsd:element name="AccessType" type="typens:FileAccess"
         minOccurs="0"/>
   </xsd:all>
</xsd:complexType>
```

Elements

Name
The name of the file.

VersionName
The version name of the file.

ReplaceExisting
Deprecated. Use Versioning instead of ReplaceExisting. Specifies whether to overwrite the latest existing version when uploading a file. If the existing file has any dependencies, BIRT iHub does not overwrite the file and creates a new version, regardless of the ReplaceExisting setting.

Versioning
Specifies what to do with the latest existing version when uploading a file. Valid values are:

- CreateNewVersion
  Always creates a new version. This is the default value.
- ReplaceLatestIfNoDependents
  Replaces the latest existing version if it does not have any dependent files. If the existing version has any dependants, BIRT iHub creates a new version instead of replacing the existing version.
- ReplaceLatestDropDependency
  Replaces the latest existing version if it does not have any dependent files. If the existing version has any dependants, BIRT iHub drops the dependency.
- ReplaceLatestMigrateDependency
  Replaces the latest existing version if it does not have any dependent files. If the existing version has any dependants, BIRT iHub moves the dependency to the new version.

MaxVersions
The maximum number of versions to keep in the Encyclopedia volume.

Description
The description of the file.

ArchiveRule
The autoarchive rules for the file.

ACL
The access rights to the file.

AccessType
The file's access type. Valid values are:

- Private
  Only the owner of the file and an administrator can access the file.

- Shared
  All users and roles specified in the access control list (ACL) for the file can access the file.

ObjectIdentifier

A complex data type that describes object identifiers.

Schema

```
<xsd:complexType name="ObjectIdentifier">
   <xsd:sequence>
      <xsd:element name="Id" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Name" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Type" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Version" type="xsd:long" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Id
The ID of the object.

Name
The name of the object.

Type
The object type.

Version
The object version number.

OpenServerOptions

A complex data type that describes open server options.

Schema

```
<xsd:complexType name="OpenServerOptions">
   <xsd:sequence>
      <xsd:element name="KeepWorkingSpace" type="xsd:boolean"/>
      <xsd:element name="DriverTimeout" type="xsd:long"/>
      <xsd:element name="PollingInterval" type="xsd:long"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

KeepWorkingSpace
Specifies whether the workspace directory is removed after the job completes.

DriverTimeout
The time for the driver to return from executing a job.

PollingInterval
The time interval for the open server to get status messages. The minimum value is 10 seconds.

PageIdentifier

A complex data type that describes page numbers.

Schema

```
<xsd:complexType name="PageIdentifier">
   <xsd:sequence>
      <xsd:element name="Range" type="xsd:string" minOccurs="0" />
      <xsd:element name="PageNum" type="xsd:long" minOccurs="0" />
      <xsd:element name="ViewMode" type="xsd:int" minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Elements

Range
A page range.

PageNum
A page number.

ViewMode
The page viewing mode.

ParameterDefinition

A complex data type that defines a report parameter.

Schema

```
<xsd:complexType name="ParameterDefinition">
   <xsd:all>
      <xsd:element name="Group" type="xsd:string" minOccurs="0"/>
      <xsd:element name="CascadingParentName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="Position" type="xsd:int" minOccurs="0"/>
      <xsd:element name="DataType" type="typens:DataType"
         minOccurs="0"/>
      <xsd:element name="DefaultValue" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="DefaultValueIsNull" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="IsRequired" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="IsPassword" type="xsd:boolean"
         minOccurs="0"/>
```

```
      <xsd:element name="IsHidden" type="xsd:boolean"
        minOccurs="0"/>
      <xsd:element name="DisplayName" type="xsd:string"
        minOccurs="0"/>
      <xsd:element name="HelpText" type="xsd:string"
        minOccurs="0"/>
      <xsd:element name="IsAdHoc" type="xsd:boolean"
        minOccurs="0"/>
      <xsd:element name="ColumnName" type="xsd:string"
        minOccurs="0"/>
      <xsd:element name="ColumnType" type="typens:DataType"
        minOccurs="0"/>
      <xsd:element name="SelectValueList"
        type="typens:ArrayOfString" minOccurs="0"/>
      <xsd:element name="SelectNameValueList"
        type="typens:ArrayOfNameValuePair" minOccurs="0"/>
      <xsd:element name="ControlType" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="AutoSuggest"/>
            <xsd:enumeration value="ControlRadioButton"/>
            <xsd:enumeration value="ControlList"/>
            <xsd:enumeration value="ControlListAllowNew"/>
            <xsd:enumeration value="ControlCheckBox"/>
            <xsd:enumeration value="FilterSimple"/>
            <xsd:enumeration value="FilterAdvanced"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="OperatorList"
          type="typens:ArrayOfString" minOccurs="0"/>
      <xsd:element name="RecordDefinition"
        type="typens:ArrayOfFieldDefinition" minOccurs="0"/>
      <xsd:element name="DefaultTableValues"
        type="typens:ArrayOfRecord" minOccurs="0"/>
      <xsd:element name="DataSourceType"
        type="typens:DataSourceType" minOccurs="0"/>
      <xsd:element name="IsViewParameter" type="xsd:boolean"
        minOccurs="0"/>
      <xsd:element name="AutoSuggestThreshold" type="xsd:long"
        minOccurs="0"/>
      <xsd:element name="IsDynamicSelectionList" type="xsd:boolean"
      minOccurs="0"/>
    </xsd:all>
  </xsd:complexType>
```

Elements

Group
The parameter group.

CascadingParentName
The cascading parent name for this parameter definition.

Name
The parameter name.

Position
The index location of the parameter in the information object (.iob) or data source map (.sma) file. The index is 1-based. If this is a regular parameter, do not specify a value or specify 0.

DataType
The data type of the parameter. Valid values are:

- Currency
- Date
- Double
- Integer
- String
- Boolean
- Structure
- Table

DefaultValue
The default value of the parameter.

DefaultValueIsNull
Flag indicating the default value is null.

IsRequired
Specifies whether the parameter is required.

IsPassword
Specifies whether a password is required.

IsHidden
Specifies whether the parameter is hidden.

DisplayName
The display name of the parameter.

HelpText
The text to display when the user holds the cursor over a parameter. For example, a value of a data column.

IsAdHoc
Specifies whether the parameter is ad hoc.

ColumnName
The name of the column on which the ad hoc parameter operates. Required for operations that include a Query element with IsAdHoc set to True, ignored otherwise.

ColumnType
The type of the column on which the ad hoc parameter operates. Required for operations that include a Query element with IsAdHoc set to True, ignored otherwise.

SelectValueList
The list of available parameter values.

SelectNameValueList
The list of available parameter names.

ControlType
The type of control used to represent the parameter. Valid values are:

- AutoSuggest
  An autosuggest control
- ControlRadioButton
  A radio button
- ControlList
  A drop-down list
- ControlListAllowNew
  A text box
- ControlCheckBox
  A check box
- FilterSimple
  A simple filter
- FilterAdvanced
  An advanced filter

OperatorList
Contains the operators used with ad hoc parameters.

RecordDefinition
The name and value of the field in the table. Used for a table parameter.

DefaultTableValues
The default values of table rows. Used for a table parameter.

DataSourceType
The type of file in which the parameter exists. Valid values are:

- ABInfoObject
  An Actuate Basic information object. This is the default value.
- InfoObject
  An information object.

IsViewParameter
Whether the parameter is a view parameter. The default value is False.

AutoSuggestThreshold
The minimum number of characters to be entered before the AutoSuggest selection list is displayed.

IsDynamicSelectionList
Flag indicating whether the selection list is dynamic or static.

ParameterValue

A complex data type that defines the value of a report parameter.

Schema

```
<xsd:complexType name="ParameterValue">
   <xsd:all>
      <xsd:element name="Group" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="DipslayName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Position" type="xsd:int" minOccurs="0"/>
      <xsd:element name="Value" type="xsd:string" minOccurs="0"/>
      <xsd:element name="ValueIsNull" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="PromptParameter" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="TableValue" type="typens:ArrayOfRecord"
         minOccurs="0"/>
      <xsd:element name="DataSourceType"
         type="typens:DataSourceType" minOccurs="0"/>
      <xsd:element name="IsViewParameter" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:all>
</xsd:complexType>
```

Elements

Group
The parameter group.

Name
The parameter name.

DisplayName
The label or display name for the parameter that appears in the user interface.

Position
The index location of the parameter in the information object (.iob) or data source map (.sma) file. The index is 1-based. If this is a regular parameter, do not specify a value or specify 0.

Value
The parameter value. Specification of both TableValue and Value causes TableValue to take precedence over Value.

ValueIsNull
A flag indicating a null value.

PromptParameter
Allows the user to select the parameter.

TableValue
The value of a table parameter. Specification of both TableValue and Value causes TableValue to take precedence over Value.

DataSourceType
The type of file in which the parameter exists. Valid values are:

- ABInfoObject
  An Actuate Basic information object. This is the default value.
- InfoObject
  An information object.

PendingSyncJob

A complex data type that describes a job in the queue waiting for Factory processing.

Schema

```
<xsd:complexType name="PendingSyncJob">
   <xsd:sequence>
      <xsd:element name="ConnectionHandle"
         type="xsd:base64binary">
      <xsd:element name="ObjectId" type="xsd:string"/>
      <xsd:element name="IsTransient" type="xsd:boolean"/>
```

```
        <xsd:element name="Volume" type="xsd:string"/>
        <xsd:element name="ServerName" type="xsd:string"
           minOccurs="0"/>
        <xsd:element name="Owner" type="xsd:string" minOccurs="0"/>
        <xsd:element name="ExecutableFileName" type="xsd:string"
           minOccurs="0"/>
        <xsd:element name="ExecutableVersionNumber" type="xsd:long"
           minOccurs="0"/>
        <xsd:element name="ExecutableVersionName" type="xsd:string"
           minOccurs="0"/>
        <xsd:element name="ResourceGroup" type="xsd:string"
           minOccurs="0" />
        <xsd:element name="SubmissionTime" type="xsd:dateTime"
           minOccurs="0"/>
        <xsd:element name="PendingTime" type="xsd:long"
           minOccurs="0"/>
        <xsd:element name="QueueTimeout" type="xsd:long"
           minOccurs="0"/>
        <xsd:element name="QueuePosition" type="xsd:long"
           minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
```

Elements

ConnectionHandle
An optional element that supports keeping a connection open to view a persistent report. If ConnectionHandle is present in the SOAP header, the system routes subsequent viewing requests to the same View service that returned the ConnectionHandle. If present, BIRT iHub System ignores the value of TargetVolume.

ObjectId
The ID of the synchronous report for which to retrieve information.

IsTransient
True if the synchronous report is transient, False if the synchronous report is persistent.

Volume
The Encyclopedia volume on which the job originated.

ServerName
The node on which the job is pending.

Owner
The name of the user who submitted the job.

ExecutableFileName
The fully qualified name of the report executable file.

ExecutableVersionNumber
The fully qualified version number of the report executable file.

ExecutableVersionName
The fully qualified version name of the report executable file.

SubmissionTime
The time at which the job was submitted to the server.

PendingTime
The number of seconds elapsed since the job entered the queue.

QueueTimeout
The number of seconds remaining before the job is deleted from the queue.

QueuePosition
The job's position in the queue.

Permission

A complex data type that describes a user's or role's privileges.

Schema

```
<xsd:complexType name="Permission">
   <xsd:sequence>
      <xsd:sequence>
         <xsd:choice minOccurs="0">
            <xsd:element name="RoleName" type="xsd:string"/>
            <xsd:element name="UserName" type="xsd:string"/>
         </xsd:choice>
         <xsd:choice minOccurs="0">
            <xsd:element name="RoleId" type="xsd:string"/>
            <xsd:element name="UserId" type="xsd:string"/>
         </xsd:choice>
      </xsd:sequence>
      <xsd:element name="AccessRight" type="xsd:string">
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
```

Elements

RoleName
The role name.

UserName
The user name.

RoleId
The role ID.

UserId
The user ID.

AccessRight
The privileges the user or role has on an object. One or more of the following characters representing a privilege:

- D—Delete
- E—Execute
- G—Grant
- V—Visible
- S—Secured Read
- R—Read
- W—Write

Printer

A complex data type that describes a printer.

Schema

```
<xsd:complexType name="Printer">
   <xsd:all>
      <xsd:element name="Name" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Manufacturer" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Model" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Location" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Description" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="SupportOrientation" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="Orientation" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="OrientationOptions"
         type="typens:ArrayOfString"
         minOccurs="0"/>
      <xsd:element name="SupportPageSize" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="PageSize" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="PageSizeOptions"
         type="typens:ArrayOfString"
         minOccurs="0"/>
```

```
        <xsd:element name="SupportScale" type="xsd:boolean"
           minOccurs="0"/>
        <xsd:element name="Scale" type="xsd:long" minOccurs="0"/>
        <xsd:element name="ScaleOptions" type="typens:ArrayOfInteger"
           minOccurs="0"/>
        <xsd:element name="SupportResolution" type="xsd:boolean"
           minOccurs="0"/>
        <xsd:element name="Resolution" type="xsd:string"
           minOccurs"=0"/>
        <xsd:element name="ResolutionOptions"
           type="typens:ArrayOfString" minOccurs="0"/>
        <xsd:element name="SupportNumberOfCopies" type="xsd:boolean"
           minOccurs="0"/>
        <xsd:element name="NumberOfCopies" type="xsd:long"
           minOccurs="0"/>
        <xsd:element name="SupportCollation" type="xsd:boolean"
           minOccurs="0"/>
        <xsd:element name="Collation" type="xsd:boolean"
           minOccurs="0"/>
        <xsd:element name="SupportPaperTray" type="xsd:boolean"
           minOccurs="0"/>
        <xsd:element name="PaperTray" type="xsd:boolean"
           minOccurs="0"/>
        <xsd:element name="PaperTrayOptions"
           type="typens:ArrayOfString" minOccurs="0"/>
        <xsd:element name="SupportDuplex" type="xsd:boolean"
           minOccurs="0"/>
        <xsd:element name="Duplex" type="xsd:string" minOccurs="0"/>
        <xsd:element name="DuplexOptions" type="typens:ArrayOfString"
           minOccurs="0"/>
        <xsd:element name="SupportColorMode" type="xsd:boolean"
           minOccurs="0"/>
        <xsd:element name="ColorMode" type="xsd:boolean"
           minOccurs="0"/>
        <xsd:element name="ColorModeOptions"
           type="typens:ArrayOfString" minOccurs="0"/>
        <xsd:element name="Status" type="xsd:string" minOccurs="0"/>
           </xsd:all>
    </xsd:complexType>
```

Elements

Name
The name of the printer. Cannot exceed 50 characters.

Manufacturer
The manufacturer of the printer.

Model
The model of the printer.

Location
The location of the printer.

Description
The description of the printer.

SupportOrientation
Specifies whether the printer supports setting paper orientation.

Orientation
The setting of the printer's orientation property.

OrientationOptions
The setting of the printer's orientation options.

SupportPageSize
Specifies whether page size can be set on the printer.

PageSize
The setting of the printer's page size property.

PageSizeOptions
The page sizes the printer supports.

SupportScale
Specifies whether the printer supports setting the scaling factor.

Scale
The setting of the printer's scaling factor.

ScaleOptions
The setting of the printer's scaling options.

SupportResolution
Specifies whether the printer supports setting the resolution.

Resolution
The setting of the printer's resolution property.

ResolutionOptions
The setting of the printer's resolution options.

SupportNumberOfCopies
Specifies whether the printer supports setting the number of copies.

NumberOfCopies
The setting of the number of copies property.

SupportCollation
Specifies whether the printer supports setting the collation.

Collation
The setting of the printer's collation property.

SupportPaperTray
Specifies whether the printer supports setting the paper tray.

PaperTray
The setting of the printer's paper tray property.

PaperTrayOptions
The setting of the printers's paper tray options.

SupportDuplex
Specifies whether the printer supports duplex printing.

Duplex
The setting of the printer's duplex property.

DuplexOptions
The setting of the printer's duplex options.

SupportColorMode
Specifies whether the printer supports printing in color.

ColorMode
The setting of the printer's color mode property.

ColorModeOptions
The setting of printer's color mode options.

Status
Indicates printer's availability.

PrinterOptions

A complex data type that describes printer options.

Schema

```
<xsd:complexType name="PrinterOptions">
   <xsd:sequence>
      <xsd:element name="PrinterName" type="xsd:string"/>
      <xsd:element name="Orientation" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="PageSize" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Scale" type="xsd:long" minOccurs="0"/>
      <xsd:element name="Resolution" type="xsd:string"
         minOccurs="0"/>
```

```
      <xsd:element name="NumberOfCopies" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="CollationOption" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="PaperTray" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Duplex" type="xsd:string" minOccurs="0"/>
      <xsd:element name="IsColor" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="IsDefaultPrinter" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

PrinterName
The name of the printer.

Orientation
The paper orientation.

PageSize
The page size.

Scale
The scaling factor.

Resolution
The resolution.

NumberOfCopies
The number of copies.

CollationOption
Turns collation on and off.

PaperTray
The paper tray.

Duplex
Sets duplex printing.

IsColor
Specifies whether the printer can print in color.

IsDefaultPrinter
Specifies whether the printer is the default printer.

PrivilegeFilter

A complex data type that represents a privilege filter. Use PrivilegeFilter to retrieve only the data accessible to roles or users with the specified privileges and to determine whether a user or role has the specified privileges on an item.

Schema

```
<xsd:complexType name="PrivilegeFilter">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="GrantedUserName" type="xsd:string"/>
         <xsd:element name="GrantedUserId" type="xsd:string"/>
         <xsd:element name="GrantedIRoleName" type="xsd:string"/>
         <xsd:element name="GrantedRoleId" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="AccessRights" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

GrantedUserName
The name of the user whose privileges to retrieve.

GrantedUserId
The ID of the user whose privileges to retrieve.

GrantedRoleName
The name of the role whose privileges to retrieve.

GrantedRoleId
The ID of the role whose privileges to retrieve.

AccessRights
The privileges.

PropertyValue

A complex data type that specifies a name-value pair.

Schema

```
<xsd:complexType name="PropertyValue">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:string" />
      <xsd:element name="Value" type="xsd:string" minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Elements

Name
The name of the property.

Value
The value of the property.

Range

A complex data type that specifies a start and end range for a search.

Schema

```
<xsd:complexType name="Range">
   <xsd:sequence>
      <xsd:element name="Start" type="xsd:long"/>
      <xsd:element name="End" type="xsd:long"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Start
The start range for the search.

End
The end range for the search.

Repeat

A complex data type that describes how often a job run is to be repeated.

Schema

```
<xsd:complexType name="Repeat">
   <xsd:sequence>
      <xsd:element name="StartTime" type="xsd:string" />
      <xsd:element name="EndTime" type="xsd:string" />
      <xsd:element name="IntervalInSeconds" type="xsd:long" />
   </xsd:sequence>
</xsd:complexType>
```

Elements

StartTime
The time that the job is to start repeatedly running.

EndTime
The time that the job is to no longer run.

IntervalInSeconds
The time to wait between job runs.

Record

A complex data type that represents a table parameter.

Schema

```
<xsd:complexType name="Record">
   <xsd:sequence>
      <xsd:element name="FieldValue" type="typens:FieldValue
         maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

FieldValue
The value of the table parameter.

ReportParameterType

A simple data type that describes parameter types.

Schema

```
<xsd:simpleType name="ReportParameterType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Execution" />
      <xsd:enumeration value="All" />
      <xsd:enumeration value="View" />
   </xsd:restriction>
</xsd:simpleType>
```

Elements

Execution
An execution parameter

View
A view parameter.

All
An all parameter type.

ResourceGroup

A complex data type the describes a resource group.

Schema

```
<xsd:complexType name="ResourceGroup">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:string"/
      <xsd:element name="Disabled" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="Description" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Type" type="xsd:string" minOccurs="0"/>
      <xsd:element name="ReportType" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Volume" type="xsd:string" minOccurs="0"/>
```

```
      <xsd:element name="MinPriority" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="MaxPriority" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="Reserved" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="StartArguments" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="WorkUnitType" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Name
The name of the resource group.

Disabled
Specifies whether the resource group can run jobs. If True, resource group does not run jobs. The default value is False.

Description
The description of the resource group.

Type
The type of jobs the resource group runs. Valid values are:

- Sync
  The resource group runs synchronous jobs.
- Async
  The resource group runs asynchronous jobs.

ReportType
The type of report the resource group creates.

Volume
The name of an Encyclopedia volume to which to assign the resource group. Valid values are:

- An empty string
  Assigns all Encyclopedia volumes on the BIRT iHub
- A volume name
  Assigns the specified Encyclopedia volume

MinPriority
Applies only to an asynchronous resource group. Specifies the minimum priority for the resource group. Valid values are 0–1,000, where 1, 000 is the highest priority. MinPriority must be less than MaxPriority. The default value is 0.

MaxPriority
Applies only to an asynchronous resource group. Specifies the maximum priority for the resource group. Valid values are 0–1,000, where 1, 000 is the highest priority. MaxPriority must be more than MinPriority. The default value is 1,000.

Reserved
Applies only to a synchronous resource group. True reserves the resource group to run only the jobs assigned to it. Use the TargetResourceGroup element in the SOAP header of an ExecuteReport request to assign a job.

StartArguments
The starting arguments for the resource group.

WorkUnitType
The license option type. An aggregate licensing model that defines iHub System features in terms of work units.

ResourceGroupSettings

A complex data type that describes the settings of a resource group.

Elements

```
<xsd:complexType name="ResourceGroupSettings">
   <xsd:sequence>
      <xsd:element name="TemplateName" type="xsd:string"/>
      <xsd:element name="Activate" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="MaxFactory" type="xsd:int" minOccurs="0"/>
      <xsd:element name="MinFactory" type="xsd:int" minOccurs="0"/>
      <xsd:element name="FileTypes" type="typens:ArrayOfString"
         minOccurs="0"/>
      <xsd:element name="StartArguments" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

TemplateName
The name of the BIRT iHub template on which the resource group runs.

Activate
Specifies whether the BIRT iHub is a member of the resource group. If True, the BIRT iHub is a member of the resource group. The default value is False.

MaxFactory
The maximum number of Factory processes available to the resource group.

MinFactory
The minimum number of Factory processes available to the resource group.

FileTypes
The file types the resource group can run.

StartArguments
The starting arguments for the resource group.

ResultSetSchema

A complex data type that describes the result set schema.

Elements

```
<xsd:complexType name="ResultSetSchema">
   <xsd:sequence>
   <xsd:element name="ResultSetName" type="xsd:string"
      minOccurs="0"/>
   <xsd:element name="ResultSetDisplayName" type="xsd:string"
      minOccurs="0"/>
   <xsd:element name="ArrayOfColumnSchema"
      type="typens:ArrayOfColumnSchema" minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
```

Elements

ResultSetName
Name of the result set.

ArrayOfColumnSchema
An array of ColumnSchema objects.

ResultSetDisplayName
The display name of the result set. If not specified, the value of the Name element is used. If the query is performed on an information object (.iob) or data source map (.sma) file, DisplayName is used as the group label.

RetryOptions

A complex data type that describes how to retry report generation or printing jobs that have failed.

Schema

```
<xsd:complexType name="RetryOptions">
   <xsd:sequence>
      <xsd:element name="RetryOption"
         type="typens:RetryOptionType"/>
      <xsd:element name="MaxRetryCount" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="RetryInterval" type="xsd:long"
         minOccurs="0"/>
```

```
   </xsd:sequence>
</xsd:complexType>
```

Elements

RetryOption
The retry options.

MaxRetryCount
The maximum number or retry attempts.

RetryInterval
The interval between retry attempts. Measured in seconds.

RetryOptionType

A simple data type that describes a retry option for failed jobs.

Schema

```
<xsd:simpleType name=”RetryOptionType”>
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Disabled"/>
      <xsd:enumeration value="VolumeDefault"/>
      <xsd:enumeration value="Specified"/>
   </xsd:restriction>
</xsd:simpleType>
```

Role

A complex data type that describes a role.

Schema

```
<xsd:complexType name="Role">
   <xsd:sequence>
      <xsd:element name="Id" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Name" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Description" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Id
The role ID.

Name
The name of the role. Cannot exceed 50 characters.

Description
The description of the role. Cannot exceed 500 characters.

RoleCondition

A complex data type that represents the fields on which a search can be performed and the condition to match.

Schema

```
<xsd:complexType name="RoleCondition">
   <xsd:sequence>
      <xsd:element name="Field" type="typens:RoleField"/>
      <xsd:element name="Match" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Field
The fields on which a search can be performed.

Match
The condition to match. If the condition includes special characters, for example a dash, either escape the special character with a backslash (\) or enclose it in brackets ([]). For example, to search for a file 05-25-04Report.rptdocument, specify one of the following:

```
05\ -25\ -04Report.rptdocument
05[-]25[-]04Report.rptdocument
```

RoleField

A simple data type that describes the fields within a role.

Schema

```
<xsd:simpleType>
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Name"/>
      <xsd:enumeration value="Description"/>
   </xsd:restriction>
</xsd:simpleType>
```

RoleSearch

A complex data type that represents a role search.

```
Schema
<xsd:complexType name="RoleSearch">
   <xsd:sequence>
      <xsd:choice minOccurs="0">
         <xsd:element name="Condition"
            type="typens:RoleCondition"/>
```

```
            <xsd:element name="ConditionArray"
               type="typens:ArrayOfRoleCondition"/>
         </xsd:choice>
         <xsd:choice minOccurs="0">
            <xsd:element name="ParentRoleName" type="xsd:string"/>
            <xsd:element name="ChildRoleName" type="xsd:string"/>
            <xsd:element name="WithRightsToChannelName"
               type="xsd:string"/>
            <xsd:element name="AssignedToUserName" type="xsd:string"/>
            <xsd:element name="ParentRoleId" type="xsd:string"/>
            <xsd:element name="ChildRoleId" type="xsd:string"/>
            <xsd:element name="WithRightsToChannelId"
               type="xsd:string"/>
            <xsd:element name="AssignedToUserId" type="xsd:string"/>
         </xsd:choice>
         <xsd:element name="FetchSize" type="xsd:int" minOccurs="0"/>
         <xsd:element name="FetchDirection" type="xsd:boolean"
            minOccurs="0"/>
         <xsd:element name="CountLimit" type="xsd:int" minOccurs="0"/>
         <xsd:element name="FetchHandle" type="xsd:string"
            minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
```

Elements

Condition
The search condition.

ConditionArray
An array of search conditions.

ParentRoleName
The name of the parent role.

ChildRoleName
The name of the child role.

WithRightsToChannelName
The name of the channel to which the role has access rights.

AssignedToUserName
The name of the user assigned to the role.

ParentRoleId
The ID of the parent role.

ChildRoleId
The ID of the child role.

WithRightsToChannelId
The ID of the channel to which the role has access rights.

AssignedToUserId
The ID of the user assigned to the role.

FetchSize
The maximum number of records to retrieve and return in a result set. The default value is 500.

FetchDirection
If True, records are retrieved forward. If False, records are retrieved backward. The default value is True.

CountLimit
The maximum number of records to count. By default, CountLimit is equal to FetchSize. To count all records, set CountLimit to -1.

FetchHandle
Retrieves more items from the result set. In the second and subsequent calls for data, specify the same search criteria as in the original call.

RunningJob

A complex data type that describes a job the Factory is currently processing.

Schema

```
<xsd:complexType name="RunningJob">
   <xsd:sequence>
      <xsd:element name="IsSyncJob" type="xsd:boolean"/>
      <xsd:element name="ConnectionHandle" type="xsd:base64Binary"
         minOccurs="0"/>
      <xsd:element name="ObjectId" type="xsd:string"
      minOccurs="0"/>
      <xsd:element name="IsTransient" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="IsProgressive" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="JobId" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Volume" type="xsd:string"/>
      <xsd:element name="ServerName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Owner" type="xsd:string" minOccurs="0"/>
      <xsd:element name="ExecutableFileName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ExecutableVersionNumber" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="ExecutableVersionName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ResourceGroup" type="xsd:string"
         minOccurs="0"/>
```

```
      <xsd:element name="SubmissionTime" type="xsd:dateTime"
         minOccurs="0"/>
      <xsd:element name="StartTime" type="xsd:dateTime"
         minOccurs="0"/>
      <xsd:element name="RunningTime" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="ExecutionTimeout" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="IsSyncFactory" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="FactoryPid" type="xsd:long"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

IsSyncJob
Specifies whether the job is synchronous. True if the job is synchronous, False if the job is asynchronous.

ConnectionHandle
An optional element that supports keeping a connection open to view a persistent report. If ConnectionHandle is present in the SOAP header, the system routes subsequent viewing requests to the same View service that returned the ConnectionHandle. If present, BIRT iHub System ignores the value of TargetVolume.

ObjectId
The ID of the synchronous report for which to retrieve information.

IsTransient
Specifies whether the synchronous report is transient. True if the synchronous report is transient, False if the synchronous report is persistent.

IsProgressive
Specifies whether progressive viewing is enabled. True if progressive viewing is enabled.

JobId
The ID of the asynchronous job.

Volume
The Encyclopedia volume on which the job originated.

ServerName
The node on which the job is running.

Owner
The name of the user who submitted the job.

ExecutableFileName
The fully qualified name of the executable file.

ExecutableVersionNumber
The version number of the executable file.

ExecutableVersionName
The version name of the executable file.

ResourceGroup
The resource group for the job.

SubmissionTime
The time at which the job was submitted to the server.

StartTime
The time at which job execution started.

RunningTime
The time elapsed since job execution started.

ExecutionTimeout
The number of seconds remaining before job execution times out. The number is always zero (infinite) for asynchronous reports.

IsSyncFactory
Specifies whether the Factory is running synchronous jobs. True if the Factory is running synchronous jobs, False if the Factory is running asynchronous jobs.

FactoryPid
The process ID of the Factory.

ScalarDataType

A simple data type that specifies a scalar parameter.

Schema

```
<xsd:simpleType name="ScalarDataType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Currency"/>
      <xsd:enumeration value="Date"/>
      <xsd:enumeration value="DateOnly"/>
      <xsd:enumeration value="Time"/>
      <xsd:enumeration value="Double"/>
      <xsd:enumeration value="Integer"/>
      <xsd:enumeration value="String"/>
      <xsd:enumeration value="Boolean"/>
   </xsd:restriction>
</xsd:simpleType>
```

Elements

Currency
A Currency parameter.

Date
A Date parameter.

DateOnly
A DateOnly parameter.

Time
A Time parameter.

Double
A Double parameter.

Integer
An Integer parameter.

String
A String parameter.

Boolean
A Boolean parameter.

SearchReportByIdList

A complex data type that describes what items to search for within a report.

Schema

```
<xsd:complextType="SearchReportByIdList">
   <xsd:sequence>
      <xsd:element name="SelectByIdList"
         type="typens:ArrayOfString"/>
      <xsd:element name="SearchByIdList"
            type="typens:ArrayOfComponent" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

SearchReportByIdList
The list of report IDs to search. Specify one of the following items:

- SelectByIdList
  The list of report IDs to select
- SearchByIdList
  The list of report IDs and values to search

SearchReportByIdNameList

A complex data type that describes what items to search for within a report.

Schema

```
<xsd:complexType name="SearchReportByIdNameList">
   <xsd:sequence>
      <xsd:element name="SelectList"
         type="typens:ArrayOfComponentIdentifier"/>
      <xsd:element name="SearchList" type="typens:ArrayOfComponent"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

SearchReportByIdNameList
The list of reports to search. Use for creating a search list in which some components are identified by ID and others are identified by name. Specify one of the following items:

- SelectList
  The list of reports to select
- SearchList
  The list of reports to search

SearchReportByNameList

A complex data type that describes what items to search for within a report.

Schema

```
<xsd:complexType name="SearchReportByNameList">
   <xsd:sequence>
      <xsd:element name="SelectByNameList"
         type="typens:ArrayOfString"/>
      <xsd:element name="SearchByNameList"
         type="typens:ArrayOfComponent" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

SearchReportByNameList
The list of report names to search. Specify one of the following items:

- SelectByNameList
  The list of report names to select
- SearchByNameList
  The list of report names and values to search

SearchResultProperty

A complex data type that describes search results.

Schema

```
<xsd:complexType name="SearchResultProperty">
   <xsd:all>
      <xsd:element name="EnableColumnHeaders" type="xsd:boolean"
         minOccurs="0" />
      <xsd:element name="UseQuoteDelimiter" type="xsd:boolean"
         minOccurs="0" />
   </xsd:all>
</xsd:complexType>
```

Elements

EnableColumnHeaders
Flag indicating whether column headers are enabled.

UseQuoteDelimiter
Flag indicating whether quotes are used as the delimiter.

ServerInformation

A complex data type that describes a BIRT iHub.

Schema

```
<xsd:complexType name="ServerInformation">
   <xsd:sequence>
      <xsd:element name="ServerName" type="xsd:string"/>
      <xsd:element name="TemplateName" type="xsd:string" />
      <xsd:element name="ServerStatusInformation"
      type="typens:ServerStatusInformation"/>
      <xsd:element name="ServiceList"
         type="typens:ArrayOfService"/>
      <xsd:element name="OwnsVolume" type="xsd:boolean"/>
      <xsd:element name="Description" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ServerVersionInformation"
         type="typens:ServerVersionInformation"/>
      <xsd:element name="ChangesPending" type="xsd:string"/>
      <xsd:element name="NodeLockViolation" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="NodeLockViolationExpirationDate"
         type="xsd:string" minOccurs="0"/>
      <xsd:element name="ServerIPAddress" type="xsd:string"
         minOccurs="0" />
       <xsd:element name="PmdPortNumber" type="xsd:int"
         minOccurs="0" />
```

```
        <xsd:element name="LocalServer" type="xsd:boolean"
          minOccurs="0" />
    </xsd:sequence>
</xsd:complexType>
```

Elements

ServerName
The name of the BIRT iHub.

TemplateName
The name of the BIRT iHub configuration template.

ServerStatusInformation
The status of the BIRT iHub. Valid values are:

- ServerState
- SystemType
- StatusErrorCode
- StatusErrorDescription

ServiceList
The list of available services.

OwnsVolume
True if there are any volumes on the BIRT iHub, False otherwise.

Description
The description of the BIRT iHub.

ServerVersionInformation
The following information about the BIRT iHub version:

- ServerVersion
- ServerBuild
- OSVersion

ChangesPending
Server configuration has changed and the BIRT iHub or the system must be restarted for the changes to take effect.

NodeLockViolation
Specifies whether a licensing node-lock violation exists. The default value is False.

NodeLockViolationExpirationDate
The date on which the grace period for a node-lock violation expires and the node lock takes effect. Contact Actuate Licensing about a node-lock licensing problem.

ServerIPAddress
The name of the BIRT iHub configuration template.

PmdPortNumber
The port where the Process Management Daemon (PMD) listens.

LocalServer
The local name of BIRT iHub.

ServerResourceGroupSetting

A complex data type that describes the settings of a resource group available to a BIRT iHub.

Schema

```
<xsd:complexType name="ServerResourceGroupSetting">
   <xsd:sequence>
      <xsd:element name="ResourceGroupName" type="xsd:string"/>
      <xsd:element name="Activate" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="Type" type="xsd:string" minOccurs="0"/>
      <xsd:element name="MaxFactory" type="xsd:int"
         minOccurs="0"/>
      <xsd:element name="MinFactory" type="xsd:int" minOccurs="0"/>
      <xsd:element name="FileTypes" type="typens:ArrayOfString"
         minOccurs="0"/>
      <xsd:element name="StartArguments" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

ResourceGroupName
The name of the resource group.

Activate
Specifies whether the BIRT iHub is a member of the resource group. If True, the BIRT iHub is a member of the resource group. The default value is False.

Type
The type of jobs the resource group runs. Valid values are:

- Sync
  The resource group runs synchronous jobs.
- Async
  The resource group runs asynchronous jobs.

MaxFactory
The maximum number of Factory processes available to the resource group.

MinFactory
The minimum number of Factory processes available to the resource group.

FileTypes
The file types the resource group can run.

StartArguments
The list of arguments used when starting a resource group process. For example, the Default Java Async resource group uses the following arguments:

```
-Xmx256M -Djava.awt.headless=true -
   Djava.protocol.handler.pkgs=com.actuate.javaserver.protocol
   com.actuate.javaserver.Server
```

ServerState

A simple data type that describes the state of an Actuate iHub.

Schema

```
<xsd:simpleType name="ServerState">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="OFFLINE"/>
      <xsd:enumeration value="STARTING"/>
      <xsd:enumeration value="ONLINE"/>
      <xsd:enumeration value="STOPPING"/>
      <xsd:enumeration value="FAILED"/>
   </xsd:restriction>
</xsd:simpleType>
```

Elements

Offline
The server is offline.

Starting
The server is starting.

Online
The server is online.

Stopping
The server is stopping.

Failed
The server failed.

ServerStatusInformation

A complex data type that describes the status of an Actuate iHub.

Schema

```
<xsd:complexType name="ServerStatusInformation">
   <xsd:sequence>
      <xsd:element name="ServerState" type="ServerState"/>
      <xsd:element name="SystemType" type="SystemType"/>
```

```
      <xsd:element name="StatusErrorCode" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="StatusErrorDescription" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

ServerState
The state of the Actuate iHub. Valid values are:

- Offline
- Starting
- Online
- Stopping
- Failed

SystemType
The type of Actuate iHub, cluster or stand-alone.

StatusErrorCode
The code of the error if status is Failed.

StatusErrorDescription
The description of the error if status is Failed.

ServerVersionInformation

A complex data type that describes the version of an Actuate iHub.

Schema

```
<xsd:complexType name="ServerVersionInformation">
   <xsd:sequence>
      <xsd:element name="ServerVersion" type="xsd:string"/>
      <xsd:element name="ServerBuild" type="xsd:string"/>
      <xsd:element name="OSVersion" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

ServerVersion
The version of the Actuate iHub.

ServerBuild
The build number of the Actuate iHub.

OSVersion
The version of the operating system.

Service

A simple data type that represents a service.

Schema

```
<xsd:simpleType name="Service">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Request"/>
      <xsd:enumeration value="Viewing"/>
      <xsd:enumeration value="Generation"/>
      <xsd:enumeration value="Caching" />
      <xsd:enumeration value="Integration" />
   </xsd:restriction>
</xsd:simpleType>
```

Elements

Request
A Message Distribution service (MDS).

Viewing
A View service.

Generation
A Factory service.

Caching
A Caching service.

Integration
An Integration service.

SortColumn

A complex data type that specifies the column on which to sort a query and the sorting order.

Schema

```
<xsd:complexType name="SortColumn">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="SortOrder">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="ASC"/>
               <xsd:enumeration value="DES"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Name
The name of the column.

SortOrder
The sort order. ASC specifies ascending order and DES specifies descending order. The values are case-sensitive. Specify in upper case.

Differentiate sort order from fetch direction. Sort order specifies the order of the data returned by a query. Fetch direction specifies whether to get the next or previous set of results in a response when the result set exceeds the FetchSize.

Stream

A complex data type that represents a streamed image.

Schema

```
<xsd:complexType name="Stream">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="EmbeddedProperty" type="xsd:boolean"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Name
The name of the image.

EmbeddedProperty
Specifies whether the image is embedded.

SupportedQueryFeatures

A simple type that describes the types of queries that can be made.

Schema

```
<xsd:simpleType name="SupportedQueryFeatures">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="UI_Version_2" />
   </xsd:restriction>
</xsd:simpleType>
```

Elements

UI_Version_2
The current version number.

SystemType

A simple data type that describes the type of BIRT iHub System.

Schema

```
<xsd:simpleType name="SystemType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Cluster"/>
      <xsd:enumeration value="Standalone"/>
   </xsd:restriction>
</xsd:simpleType>
```

Elements

Cluster
A cluster system.

Standalone
A stand-alone system.

TypeName

A simple data type that describes names of data types.

Schema

```
<xsd:simpleType name="TypeName">
   <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="int" />
      <xsd:enumeration value="sht" />
      <xsd:enumeration value="dbl" />
      <xsd:enumeration value="dbn" />
      <xsd:enumeration value="cur" />
      <xsd:enumeration value="dtm" />
      <xsd:enumeration value="str" />
      <xsd:enumeration value="bln" />
      <xsd:enumeration value="nll" />
   </xsd:restriction>
</xsd:simpleType>
```

TypeName
The name of the type.

User

A complex data type that describes a user.

Schema

```
<xsd:complexType name="User">
   <xsd:sequence>
      <xsd:element name="Id" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Name" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Password" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Description" type="xsd:string"
         minOccurs="0"/>
```

```
      <xsd:element name="IsLoginDisabled" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="EmailAddress" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="HomeFolder" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ViewPreference" minOccurs="0">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="Default"/>
               <xsd:enumeration value="DHTML"/>
               <xsd:enumeration value="LRX"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="MaxJobPriority" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="SendNoticeForSuccess" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="SendNoticeForFailure" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="SuccessNoticeExpiration" type="xsd:long"
         minOccurs="0"/>
      </xsd:element>
      <xsd:element name="FailureNoticeExpiration" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="SendEmailForSuccess" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="SendEmailForFailure" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="AttachReportInEmail" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="DefaultPrinterName" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Id

The user's user ID.

Name

The user's name. A user name is a string of 1 to 256 characters, including any character except a control character. A user name is not case-sensitive. BIRT iHub stores a user name in mixed case, always displaying it exactly the way it was typed during creation.

Password
The user's password. A password is a string of 1 to 256 characters, including any character except a control character or space. Security experts recommend using passwords of at least eight characters, including mixed-case alphabetic and numeric characters. A password is case-sensitive. The Administrator can change any user's password. Users can only change their own passwords. BIRT iHub encrypts a user's password.

Description
The description of the user.

IsLoginDisabled
Specifies whether the user can log in.

EmailAddress
The user's e-mail address.

HomeFolder
The user's home folder.

ViewPreference
The user's viewer, Default or DHTML.

MaxJobPriority
The maximum priority that the user can assign to a job.

SendNoticeForSuccess
Specifies whether the BIRT iHub sends success notices to the user.

SendNoticeForFailure
Specifies whether the BIRT iHub sends failure notices to the user.

SuccessNoticeExpiration
Specifies the minimum number of minutes success notices remain in the Completed folder. After this time elapses, BIRT iHub removes the notices the next time it removes requests from the volume's Requests\Completed folder. If not set or set to 0, DefaultSuccessNoticeExpiration specified in Volume is used. To set the user's success notices to never expire, set the value to 0xffffffff.

FailureNoticeExpiration
Specifies the minimum number of minutes failure notices remain in the Completed folder. After this time elapses, BIRT iHub removes the notices the next time it removes requests from the volume's Requests\Completed folder. If not set or set to 0, DefaultFailureNoticeExpiration specified in Volume is used. To set the user's failure notices to never expire, set the value to 0xffffffff.

SendEmailForSuccess
Specifies whether the BIRT iHub sends success notices in an e-mail message to the user.

SendEmailForFailure
Specifies whether the BIRT iHub sends failure notices in an e-mail message to the user.

AttachReportInEmail
Specifies whether to attach a report to an e-mail completion notice.

DefaultPrinterName
The name of the user's default printer.

UserCondition

A complex data type that represents the fields on which a search can be performed and the condition to match.

Schema

```
<xsd:complexType name="UserCondition">
   <xsd:sequence>
      <xsd:element name="Field" type="typens:UserField">
      <xsd:element name="Match" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Field
The fields on which a search can be performed.

Match
The condition to match. If the condition includes special characters, for example a dash, either escape the special character with a backslash (\) or enclose it in brackets ([]). For example, to search for a file 05-25-04Report.rptdocument, specify one of the following:

```
05\ -25\ -04Report.rptdocument
05[-]25[-]04Report.rptdocument
```

UserField

A simple data type that describes the fields within a user element.

Schema

```
<xsd:simpleType name="UserField">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Name"/>
      <xsd:enumeration value="Description"/>
      <xsd:enumeration value="IsLoginDisabled"/>
      <xsd:enumeration value="EmailAddress"/>
      <xsd:enumeration value="HomeFolder"/>
      <xsd:enumeration value="ViewPref"/>
```

```
      <xsd:enumeration value="MaxJobPriority"/>
      <xsd:enumeration value="SuccessNoticeExpiration"/>
      <xsd:enumeration value="FailureNoticeExpiration"/>
      <xsd:enumeration value="SendNoticeForSuccess"/>
      <xsd:enumeration value="SendNoticeForFailure"/>
      <xsd:enumeration value="SendEmailForSuccess"/>
      <xsd:enumeration value="SendEmailForFailure"/>
      <xsd:enumeration value="AttachReportInEmail"/>
      <xsd:enumeration value="DefaultPrinterName"/>
   </xsd:restriction>
</xsd:simpleType>
```

UserSearch

A complex data type that represents a user search.

Schema

```
<xsd:complexType name="UserSearch">
   <xsd:sequence>
      <xsd:choice minOccurs="0">
         <xsd:element name="Condition"
            type="typens:UserCondition"/>
         <xsd:element name="ConditionArray"
            type="typens:ArrayOfUserCondition"/>
      </xsd:choice>
      <xsd:choice minOccurs="0">
         <xsd:element name="MemberOfGroupName" type="xsd:string"/>
         <xsd:element name="WithRoleName" type="xsd:string"/>
         <xsd:element name="SubscribedToChannelName"
            type="xsd:string"/>
         <xsd:element name="MemberOfGroupId" type="xsd:string"/>
         <xsd:element name="WithRoleId" type="xsd:string"/>
         <xsd:element name="WithLicenseOption" type="xsd:string"/>
         <xsd:element name="SubscribedToChannelId"
            type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="FetchSize" type="xsd:int" minOccurs="0"/>
      <xsd:element name="FetchDirection" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="CountLimit" type="xsd:int" minOccurs="0"/>
      <xsd:element name="FetchHandle" type="xsd:string"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

Condition

The search condition.

ConditionArray
An array of search conditions.

MemberOfGroupName
The name of the group of which the user is a member.

WithRoleName
The name of the role to which the user belongs.

SubscribedToChannelName
The name of the channel to which the user is subscribed.

MemberOfGroupId
The ID of the group of which the user is a member.

WithRoleId
The ID of the role to which the user belongs.

WithLicenseOption
The name of the license option assigned to the user.

SubscribedToChannelId
The ID of the channel to which the user is subscribed.

FetchSize
The maximum number of records to retrieve and return in a result set. The default value is 500.

FetchDirection
If True, records are retrieved forward. If False, records are retrieved backward. The default value is True.

CountLimit
The maximum number of records to count. By default, CountLimit is equal to FetchSize. To count all records, set CountLimit to -1.

FetchHandle
Retrieves more items from the result set. In the second and subsequent calls for data, specify the same search criteria as in the original call.

VersioningOption

A simple data type that specifies the options for handling the latest existing version when uploading a file.

Schema

```
<xsd:simpleType name="VersioningOption">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="CreateNewVersion" />
```

```
      <xsd:enumeration value="ReplaceLatestIfNoDependents" />
      <xsd:enumeration value="ReplaceLatestDropDependency" />
      <xsd:enumeration value="ReplaceLatestMigrateDependency" />
   </xsd:restriction>
</xsd:simpleType>
```

Elements

CreateNewVersion
Always creates a new version. This is the default value.

ReplaceLatestIfNoDependents
Replaces the latest existing version if it does not have any dependent files. If the existing version has any dependants, BIRT iHub creates a new version instead of replacing the existing version.

ReplaceLatestDropDependency
Replaces the latest existing version if it does not have any dependent files. If the existing version has any dependants, BIRT iHub drops the dependency.

ReplaceLatestMigrateDependency
Replaces the latest existing version if it does not have any dependent files. If the existing version has any dependants, BIRT iHub moves the dependency to the new version.

ViewParameter

A complex data type that describes a viewing parameter.

Schema

```
<xsd:complexType name="ViewParameter">
   <xsd:all>
      <xsd:element name="Format" type="xsd:string" minOccurs="0"/>
      <xsd:element name="UserAgent" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ScalingFactor" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="AcceptEncoding" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ViewOperation" minOccurs="0">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="view"/>
               <xsd:enumeration value="print"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="PathInformation" type="xsd:string"
         minOccurs="0"/>
```

```
      <xsd:element name="EmbeddedObjPath" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="RedirectPath" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="PdfQuality" type="xsd:long"
         minOccurs="0"/>
   </xsd:all>
</xsd:complexType>
```

Elements

Format

The format in which the report displays. Valid formats are:

- CSS
- DHTML
- DHTMLLong
- DHTMLRaw
- ExcelData
  Does not support specifying component ID
- ExcelDisplay
  Does not support specifying component ID
- ImageMapURL
- PDF
  Does not support specifying component ID
- PPT
- PPTFullyEditable
- Reportlet
  Valid only if ShowInReportlet is enabled during report design
- RTF
  Does not support specifying component ID
- RTFFullyEditable
  Does not support specifying component ID
- XMLCompressedDisplay
- XMLCompressedExcel
- XMLCompressedPDF

- XMLCompressedPPT
- XMLCompressedReportlet
- XMLCompressedRTF
- XMLData
- XMLDisplay
- XMLReportlet
- XMLStyle

To support users clicking a point in a chart to navigate to different report sections, set Format to ImageMapURL.

SearchReport uses a different set of formats than other operations that use the ViewParameter data type. For SearchReport, valid formats are:

- CSV
- EXCEL
- TSV
- UNCSV
- UNTSV
- XMLDisplay

UserAgent
The browser to use for report viewing, such as Mozilla/4.0.

ScalingFactor
Adapts the size of a Reportlet to the Reportlet frame.

AcceptEncoding
The list of encoding methods the browser supports.

ViewOperation
The view operation, View or Print.

PathInformation
The path to the report.

EmbeddedObjPath
The base URL to prepend to a static or dynamic object in a report. When viewing a report in a browser, the URL of an image, chart, JavaScript, or another resource refers to the Encyclopedia volume. Use EmbeddedObjPath to change this URL.

RedirectPath
Maps from the current URL to a new target.

PdfQuality
The viewing quality of a PDF.

Volume

A complex data type that describes an Encyclopedia volume.

Schema

```
<xsd:complexType name="Volume">
   <xsd:all>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="ActuateVersion" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ActuateBuildNumber" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="SecurityIntegrationOption" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="OpenSecuritySelectUsersOfRole"
         type="xsd:boolean" minOccurs="0"/>
      <xsd:element name="OpenSecuritySelectGroupsOfUser"
         type="xsd:boolean" minOccurs="0"/>
      <xsd:element name="DefaultPrinterName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="MaxJobRetryCount" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="JobRetryInterval" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="DefaultViewingPreference" minOccurs="0">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="LRX"/>
               <xsd:enumeration value="DHTML"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="DHTMLPageCaching" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="DHTMLPageCachingExpirationAge"
         type="xsd:long" minOccurs="0"/>
      <xsd:element name="IsAutoArchiveRunning" type="xsd:boolean"
         minOccurs="0"/>
         <xsd:element name="AuthorizationIsExternal"
            type="xsd:boolean" minOccurs="0"/>
```

```
      <xsd:element name="ConnectionPropertiesAreExternal"
         type="xsd:boolean" minOccurs="0"/>
      <xsd:element name="DefaultSuccessNoticeExpiration"
         type="xsd:long" minOccurs="0"/>
      <xsd:element name="DefaultFailureNoticeExpiration"
         type="xsd:long" minOccurs="0"/>
      <xsd:element name="ResourcePath" type="xsd:string"
         minOccurs="0"/>
   </xsd:all>
</xsd:complexType>
```

Elements

Name
The name of the volume.

ActuateVersion
The version number.

ActuateBuildNumber
The build number.

SecurityIntegrationOption
The security integration option.

OpenSecuritySelectUsersOfRole
Applies only if using external registration security level. Indicates whether the SelectUsers operation for a role is supported. If the operation is supported, iHub enables appropriate features in iHub Management Console.

OpenSecuritySelectGroupsOfUser
Applies only if using external registration security level. Indicates whether the SelectGroups operation for a user is supported. If the operation is supported, iHub enables appropriate features in iHub Management Console.

DefaultPrinterName
The name of the default printer.

MaxJobRetryCount
The maximum number of retry attempts.

JobRetryInterval
The interval between retry attempts. Measured in seconds.

DefaultViewingPreference
The default viewer.

DHTMLPageCaching
True enables DHTML page caching.

DHTMLPageCachingExpirationAge
If DHTMLPageCaching is True, set DHTMLPageCachingExpirationAge to a valid value. To disable DHTMLPageCaching, set DHTMLPageCachingExpirationAge to -1.

IsAutoArchiveRunning
Determines whether an archive pass is currently running. If True, an archive pass is running.

AuthorizationIsExternal
True enables external user registration.

ConnectionPropertiesAreExternal
Specifies whether connection properties are externalized using the Report Server Security Extension (RSSE).

DefaultSuccessNoticeExpiration
Specifies the minimum number of minutes success notices remain in the Completed folder. After this time elapses, BIRT iHub removes the notices the next time it removes requests from the volume's Requests\Completed folder. This time applies to all users whose SuccessNoticeExpiration time is not set or is set to 0. The default value is 0, which means notices never expire.

DefaultFailureNoticeExpiration
Specifies the minimum number of minutes failure notices remain in the Completed folder. After this time elapses, BIRT iHub removes the notices the next time it removes requests from the volume's Requests\Completed folder. This time applies to all users whose SuccessNoticeExpiration time is not set or is set to 0. The default value is 0, which means notices never expire.

ResourcePath
The resource path to the volume.

Weekly

A complex data type describing weekly job scheduling.

Schema

```
<xsd:complexType name="Weekly">
   <xsd:sequence>
      <xsd:element name="FrequencyInWeeks" type="xsd:long" />
      <xsd:element name="RunOn" type="xsd:string" />
      <xsd:element name="OnceADay" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Repeat" type="typens:Repeat"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

FrequencyInWeeks
The number of times a job is to run, in weeks.

RunOn
The day to run the job.

OnceADay
The time to run the job.

Repeat
The number of times to repeat the schedule.

Part Three

Working with BIRT iHub integration APIs

Chapter

10

Using Java Report Server Security Extension

This chapter consists of the following topics:

- About the Java Report Server Security Extension
- Implementing the Java RSSE interface
- About installing a Java RSSE application
- Using page-level security
- SOAP-based Report Server Security Extension (RSSE) operations
- SOAP-based Report Server Security Extension (RSSE) data types

About the Java Report Server Security Extension

BIRT iHub System provides a SOAP-based API that supports running a Report Server Security Extension (RSSE) application as a web service. Using the Java RSSE framework, a developer can create an application that provides one of the following security features:

- External authentication

  Authenticates a user's password using an interface to an external security system such as Lightweight Directory Access Protocol (LDAP) or Microsoft Active Directory. Users, roles, notification groups, access control lists (ACLs), and other information remain on the Encyclopedia volume.

- External registration

  Manages users, roles, notification groups, access control lists (ACLs), and other information using an interface to an external security system such as Lightweight Directory Access Protocol (LDAP) or Microsoft Active Directory. The Encyclopedia volume no longer manages this information.

- Page-level security

  Controls user access to sensitive information in a BIRT report by implementing page-level security. A page-level security application requires a BIRT Page Level Security option license.

The following sections describe how to build, install, and customize these Java RSSE security applications in the Actuate Information Delivery API development environment.

Implementing the Java RSSE interface

BIRT iHub Integration Technology provides sample applications that show how to implement the Java RSSE interface. In the installation, each sample application is located in a separate subdirectory under the Java Report Server Security Extension directory.

Each sample application provides the following resources:

- The reference implementation of the RSSE interface.

  Table 10-1 lists the package for each sample application.

Table 10-1 Sample application packages

RSSE application	Package
LDAP authentication	com.actuate11.rsse.authenticationSample

Table 10-1 Sample application packages

RSSE application	Package
LDAP external registration	com.actuate11.rsse.ldapSample
Page Security	com.actuate11.rsse.aclSample

- The file, lib/rsse.jar, contains the RSSE interface and related classes.

  The documentation for the package, com.actuate11.rsse.interfaces, is in the Java Report Server Security Extension/docs folder. To implement a class for the RSSE interface, refer to this API reference.

- The object renderer package, com.actuate11.rsse.or, contains a set of helper classes for logging RSSE objects to a file.

  The package uses the open source logging tool, Apache log4j. Using the Apache log4j API, a developer can write log statements in the application code, then configure the logging level through a property file.

  To configure the logging level for a Java RSSE application, modify the property, log4j.logger.com.actuate11.rsse, in the file, log.properties. The file, log.properties, is in the application package.

  Apache log4j supports logging at the following levels:

  - FATAL describes a severe error event that typically causes the application to abort.
  - ERROR describes an error event that typically allows the application to continue running.
  - WARN provides an alert to a potential problem.
  - INFO provides a general message that describes the application's progress.
  - DEBUG provides information on an application event that is useful for debugging.
  - ALL turns on all logging options.
  - OFF turns off logging.

  For more information about the Apache Logging Services Project and the log4j tool, see http://logging.apache.org/.

About installing a Java RSSE application

To set up and run a Java RSSE application, perform the following tasks:

- Build the Java RSSE application.

- Install the Java RSSE application on an Encyclopedia volume.
- Enable a web service to use the Java RSSE application.

How to build a Java RSSE sample application

Install and use the Apache Ant tool to build a Java RSSE sample application. Go to the Apache Ant Project web site at http://ant.apache.org/ to obtain the software and installation instructions.

In the BIRT iHub Integration Technology installation, each sample application subdirectory contains a file, build.xml. Using the project settings specified in the file, build.xml, Ant performs the following operations:

- Compiles the Java RSSE application source files
- Creates a lib directory
- Archives the compiled classes in a JAR file in the lib directory
  Table 10-2 lists the archive file generated for each Java RSSE sample application.

Table 10-2 Archive files that are generated for the Java RSSE sample applications

RSSE application	Archive file
LDAP authentication	rsseAuthenticate.jar
LDAP external registration	rsseLdap.jar
Page Security	rsseAcl.jar

To build a sample application using the Ant tool, navigate to the application directory. At the command line, type:

```
ant
```

Installing a Java RSSE application

Configure each Encyclopedia volume that runs RSSE web service applications separately. A SOAP-based RSSE application runs as a web service in the BIRT iHub servlet container.

The default location for an RSSE web service application is $SERVER_HOME /servletcontainer/webapps/acrsse. To run multiple, SOAP-based, RSSE applications on multiple Encyclopedia volumes on BIRT iHub, configure a separate location for each RSSE application.

How to install a Java RSSE application on an Encyclopedia volume

Install Java RSSE applications on Encyclopedia volumes to run on BIRT iHub by performing the following tasks:

1. Make a copy of the $SERVER_HOME/servletcontainer/webapps/acrsse directory. For example, copy the directory to the following location:

   ```
   $SERVER_HOME/servletcontainer/webapps/myacrsse
   ```

2. Copy the application archive file to the lib directory of the BIRT iHub servlet container in the following location:

   ```
   $SERVER_HOME/servletcontainer/webapps/myacrsse/WEB-INF/lib
   ```

3. Extract the file, class.properties, from the application archive file, to the following location:

   ```
   $SERVER_HOME /servletcontainer/webapps/myacrsse/WEB-INF
      /classes/com/actuate11/rsse/wsdl
   ```

   If necessary, create the subdirectories, /com/actuate11/rsse/wsdl, manually or use the archive extraction tool to create the subdirectories when extracting the class.properties file.

4. Using a source code editor, open the class.properties file and change its single line of code to reference the main class of the application in the archive file:

   ```
   class=com.actuate11.rsse.mySampleApp.SampleRSSE
   ```

Configuring and deploying an LDAP configuration file

To use a Java RSSE sample application that utilizes LDAP for external user authentication or registration, configure and deploy an LDAP configuration file to the BIRT iHub etc directory before enabling the web service on the Encyclopedia volume.

How to configure and deploy an LDAP configuration file for external authentication

To configure and deploy the LDAP configuration file for external authentication perform the following operations:

1. Using a source code editor, create the LDAP configuration file by typing the following code, substituting the values appropriate for the LDAP server installation such as:

   - Name of the LDAP server
   - Port number where the LDAP server listens
   - UserBaseDN, including the attributes for the organizational unit, ou, and domain components, dc

   ```
   <!-- ldapconfig_$volumeName.xml -->
   <!--"-->
   <Config>
      <!--The name of the LDAP server.-->
           <Server>servername.actuate.com</Server>
      <!--The port number where the LDAP server listens.-->
   ```

```
   <Port>389</Port>
   <!--The base DN used for user queries.-->
   <UserBaseDN>ou=actuate users, dc=actuate, dc=com
      </UserBaseDN>
</Config>
```

2 Save the file to the following location, naming the file, ldapconfig_$volumeName.xml, changing $volumeName to the Encyclopedia volume name:

```
\Program Files\Actuate\iHub\etc\
```

How to configure and deploy an LDAP configuration file for external registration

Install the Java RSSE external registration example on an Encyclopedia volume of BIRT iHub by performing the following tasks:

1 Using a source code editor, create an LDAP configuration file and copy or type the following code, substituting the values appropriate for the LDAP server installation:

```
<!--"-->
<Config>
   <!-- Name of the LDAP server. -->
   <Server>servername</Server>
      <!-- Port number where the LDAP server listens. The
      default port is 389. -->
   <Port>389</Port>
   <!-- LDAP distinguished name that the RSSE application uses
   for a query operation to the LDAP server. The Open Security
   application uses this account to validate users, roles, ACLs,
   and other Encyclopedia user information.  Account with READ
   privilege is sufficient. -->
   <QueryAccount>uid=admin, ou=Administrators,
      ou=TopologyManagement, o=NetscapeRoot</QueryAccount>
   <!-- Password for the LDAP account specified by the
   QueryAccount parameter. -->
   <QueryPassword>actuate</QueryPassword>

   <!-- LDAP distinguished name that the RSSE application uses
   to locate the LDAP user object, including attributes for the
   organizational unit, ou, and domain components, dc. -->
   <UserBaseDN>ou=AcUsers,dc=actuate,dc=com</UserBaseDN>

   <!-- Name of  LDAP object class that the Actuate open
   security application uses to find Actuate user names. -->
   <UserObject>inetorgperson</UserObject>

<!-- Actuate role attribute that indicates that an LDAP user
object can perform Encyclopedia volume administration. -->
   <AdminRole>AcAdmin</AdminRole>
```

```
<!-- LDAP role object name that maps to the Encyclopedia
volume Operator role. -->
<OperatorRole>AcAdmin</OperatorRole>

<!-- LDAP role object that maps to the All role in the
Encyclopedia volume. -->
<AllRole>All</AllRole>

<!-- LDAP distinguished name that the RSSE application uses
to locate the LDAP role object. -->
<RoleBaseDN>ou=AcRoles,dc=actuate, dc=com</RoleBaseDN>

<!-- LDAP object class that the Actuate open security
application uses to find Actuate role names. -->
<RoleObject>groupofuniquenames</RoleObject>

<!-- LDAP distinguished name that the RSSE application uses
to locate the LDAP Actuate notification group object.
GroupBaseDN can be the same as the role DN, if Group
information is not separately maintained. -->
<GroupBaseDN>ou=groups,dc=actuate, dc=com</GroupBaseDN>

<!-- LDAP object class that the Actuate open security
application uses to find Actuate notification group
names. -->
<GroupObject>groupofuniquenames</GroupObject>

<!-- Name of the LDAP group used for notifications of all job
requests made in the iHub. The base DN is obtained from
GroupBaseDN. -->
<GroupToNotify>specialGroup</GroupToNotify>

<!-- LDAP attribute used to retrieve the EmailAddress
property of the user. No default value. -->

<EmailAddressAttr>mail</EmailAddressAttr>

<!-- LDAP attribute used to retrieve the license option
property of the user. No default value. -->
<LicenseOptionsAttr>actuatelicenseoptions
   </LicenseOptionsAttr>

<!-- LDAP attribute used to retrieve the HomeFolder property
of the user. No default value. -->
<HomeFolderAttr>actuateHomeFolder</HomeFolderAttr>

<!-- LDAP attribute used to retrieve the AttachReportInEmail
property of the user. -->
<AttachReportInEmailAttr>actuateEmailForm
</AttachReportInEmailAttr>

<!-- Permitted values are "included" or "linked". The default
value is "linked". -->
```

```
<AttachReportInEmailDefault>linked
</AttachReportInEmailDefault>

<!-- LDAP attribute used to retrieve the email preferences,
SendEmailForSuccess and SendEmailForFailure properties of the
user.  For some object classes such as inetorgperson, an
e-mail attribute exists in the standard LDAP schema. -->
<SendEmailAttr>actuateEmailWhen</SendEmailAttr>

<!-- Permitted values are "never", "always", "failures", or
"successes". -->
<SendEmailDefault>never</SendEmailDefault>

<!-- LDAP attribute used to retrieve the notification
preferences, SendNoticeForSuccess and SendNoticeForFailure
properties of the user. -->
<SendNoticeAttr>actuateFolderWhen</SendNoticeAttr>

<!-- Permitted values are  "never", "always", "failures",
"successes". -->
<SendNoticeDefault>always</SendNoticeDefault>

<!-- LDAP attribute used to retrieve the
SuccessNoticeExpiration property of the user. The default
value causes BIRT iHub to delete notices according to volume
settings.-->
   <SuccessNoticeExpirationAttr>
   actuateSuccessNoticeExpiration
</SuccessNoticeExpirationAttr>

<!-- Value to use for SuccessNoticeExpirationAttr when LDAP
does not contain a value for that attribute. The value is the
number of minutes. The default value of 0 (zero) causes BIRT
iHub to delete notices according to volume settings. A value
of -1 means that BIRT iHub keeps notices indefinitely. -->
<SuccessNoticeExpirationDefault>0
</SuccessNoticeExpirationDefault>

<!-- LDAP attribute used to retrieve the
FailureNoticeExpiration property of the user. The default
value causes BIRT iHub to delete notices according to volume
settings. -->
<FailureNoticeExpirationAttr>actuateFailNoticeExpiration
</FailureNoticeExpirationAttr>

<!-- Value to use for FailureNoticeExpirationDefault when
LDAP does not contain a value for that attribute. The value
is the number of minutes. The default value of 0 (zero)
causes BIRT iHub to delete notices according to volume
settings. A value of -1 means that BIRT iHub keeps notices
indefinitely. -->
```

```
<FailureNoticeExpirationDefault>0
</FailureNoticeExpirationDefault>

<!-- LDAP attribute used to retrieve the privilege template
for a user. Value is a comma-separated list of user or role
privileges. A user permission is a user name followed by "="
and a string of 0 (zero) or more permission characters. A
role permission is a role name followed by a "~" and a string
of permission characters.
   Permissible characters and their meanings are:
   "r" = read
   "w" = write
   "e" = execute
   "d" = delete
   "v" = visible
   "s" = secure read (page level read)
   "g" = grant
   Examples: bob=rwed,  viewing only~rv -->
<PrivilegeTemplateAttr>actuateDefaultPriv
</PrivilegeTemplateAttr>

   <!-- Value to use for PrivilegeTemplateAttr when LDAP does
   not contain a value for that attribute. -->
   <PrivilegeTemplateDefault/>

<!-- LDAP attribute used to retrieve the MaxJobPriority
property of the user. The default value is 500. The
permissible range is 0-1000. -->
<MaxJobPriorityAttr>actuateMaxPriority
</MaxJobPriorityAttr>

<!-- Value to use for MaxJobPriority when LDAP does not
contain a value for that attribute. Default is 500. -->
<MaxJobPriorityDefault>500</MaxJobPriorityDefault>

<!-- LDAP attribute used to retrieve the ViewPreference
property of the user. -->
<ViewPreferenceAttr>actuateViewingPref
</ViewPreferenceAttr>

<!-- Value to use for ViewPreferenceAttr when LDAP does not
contain a value for that attribute. Permissible values are
"default" and "dhtml".-->
<ViewPreferenceDefault>default</ViewPreferenceDefault>
<!-- LDAP attribute used to retrieve the channel subscription
list of the user. The values are specified as a comma-separated
list. -->
<ChannelSubscriptionListAttr>actuateChannelList
</ChannelSubscriptionListAttr>
```

```
    <!-- Value to use for ChannelSubscriptionListAttr when LDAP
    does not contain a value for that attribute. The value is a
    comma-separated lists of channel names or is empty. -->
    <ChannelSubscriptionListDefault/>
    <ConnectionPropertyList>
       <ConnectionProperty>
          <Name>username</Name>
          <Value>testUser</Value>
       </ConnectionProperty>
       <ConnectionProperty>
          <Name>password</Name>
          <Value>mypassword</Value>
       </ConnectionProperty>
    </ConnectionPropertyList>
    <!--  LDAP attributes used when externalizing
    ConnectionPropertyList, containing username and password.
    Typically used when implementing pass-through security. Do
    not include the ConnectionPropertyList if not externalizing
    these properties. -->
    <ConnectionPropertyList>
       <ConnectionProperty>
          <Name>username</Name>
          <Value>testUser</Value>
       </ConnectionProperty>
       <ConnectionProperty>
          <Name>password</Name>
          <Value>mypassword</Value>
       </ConnectionProperty>
    </ConnectionPropertyList>
</Config>
```

2 Save the file to the following location, naming the file, ldapconfig_$volumeName.xml, changing $volumeName to the Encyclopedia volume name:

```
\Program Files\Actuate\iHub\etc\
```

How to prepare an Encyclopedia volume to use external user registration

To use a Java RSSE sample application that utilizes LDAP for external user registration, in the properties section of the chosen Encyclopedia volume, enable the open security web service, choose OK, and then restart the volume. Also, configure an LDAP server database to contain the Encyclopedia volume's user information. For more information about configuring the LDAP server database, see the LDAP server documentation.

How to enable the open security web service to use the Java RSSE application

To enable Open Security as a web service on an Encyclopedia volume, perform the following operations:

1 Log in to the iHub Configuration Console, choose Advanced View, and perform the following operations:

   1 Choose Volumes.

   2 In Volumes, from the Encyclopedia volume's drop-down list, choose Properties, as shown in Figure 10-1.

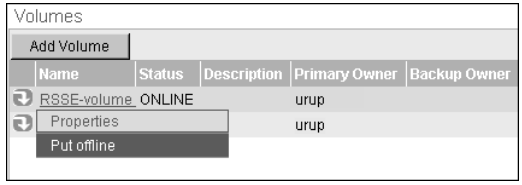


Figure 10-1 System volume properties

   3 On Volume—Properties, choose Open Security.

   4 On Open Security, in Enable/Disable, choose Enable as web service.

   In Web service, specify the following information as required:

   - IP address or machine name where the web service resides
   - SOAP port where the web service listens
   - Context string indicating the path for BIRT iHub to use when sending messages to the web service.

   Open Security looks like Figure 10-2.

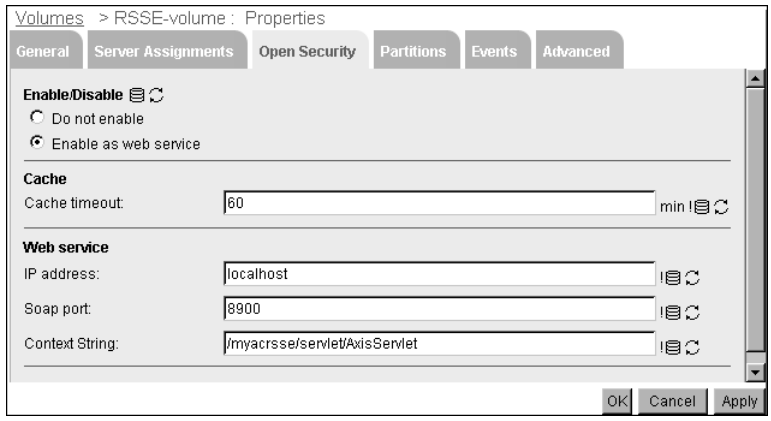


Figure 10-2 The Open Security tab

   Choose OK.

   Starting with Release 11, the default location for acserverconfig.xml and acserverlicense.xml is AC_DATA_HOME/server/config. AC_DATA_HOME refers to the folder the installer specified as the location

for data during the iHub installation. By default, that path is C:/Actuate/iHub/data on a Windows system, and /<Installation directory>/AcServer/data on a Linux system.

5 From the Encyclopedia volume's drop-down list, choose Put offline, as shown in Figure 10-3.

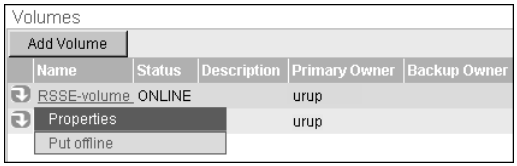


Figure 10-3 Putting an Encyclopedia volume offline

2 Take the Encyclopedia volume online.

3 Test the installation of the Java RSSE application.

For example, log in to the Encyclopedia volume using iHub Management Console as the user, Administrator, as defined in the LDAP configuration file, typing the password specified in the LDAP server.

Installing the page-level security application

To install the Java RSSE page-level security sample application on BIRT iHub, deploy an external access control list (ACL) file with the application. Perform this operation before enabling the web service on the Encyclopedia volume.

For more information about deploying the ACL file in the page-level security application installation, see "How to install the Java RSSE page-level security application," later in this chapter.

To complete the installation, perform the following steps:

1 Load a sample executable report to the Encyclopedia volume.

2 Run the executable report to create a report document.

3 Configure permissions for these report files to test the sample application installation.

Migrating a Java RSSE application to a new Actuate release

When migrating a Java RSSE application from an older release of Actuate software to a newer release, the application may require recompilation using the new release's libraries.

When modifying the software, update any references of the older release to reference the new release, and place all relevant JAR files that contain classes for the new version into their proper locations.

Using page-level security

Using the iHub Report Server Security Extension (RSSE) framework, a developer can create an RSSE service that manages page-level security in Actuate BIRT designs by retrieving a user's access control list (ACL) externally.

By default, when a secure report asks for the ACL of a user, the Encyclopedia volume returns a list that includes the user ID and the roles in which the user is a member. Frequently, the information in BIRT iHub security does not match the information in a database used by a secure design. An RSSE page security application can translate a BIRT iHub ACL to a design-specific ACL.

How to install the Java RSSE page-level security application

BIRT iHub Integration Technology contains an example of how external page-level security works using Java RSSE in the subdirectory, Page_Security_Example. For information about Actuate BIRT design page-level security, see *Using Actuate BIRT.*

To install the page-level security sample application on BIRT iHub, deploy an external ACL file with the application. Perform this operation before enabling the web service on the Encyclopedia volume.

To include the ACL file provided with the sample application in the build, perform the following operations:

1. Copy the file, user.acls, located in the Page_Security_Example directory to the following location:

   ```
   /com/actuate11/rsse/aclSample
   ```

2. Using a source code editor, in Page_Security_Example directory, open the file, build.xml, and perform the following operations:

   1. In build.xml, navigate to the buildACL element specifying the contents of the file, rsseAcl.jar.
   2. Modify the fileset list to contain the following line of code:

      ```
      <include name="com/**/*.acls" />
      ```

      The buildACL element looks like the following example:

      ```
      <target name="buildACL" depends="buildACL.clean, compileACL">
         <mkdir dir="lib"/>
      ```

```
    <jar jarfile="lib/rsseAcl.jar">
      <fileset dir=".">
        <include name="com/**/*.class" />
    <include name="com/**/*.properties" />
        <include name="com/**/*.acls" />
      </fileset>
    </jar>
  </target>
```

3 Build the application using the Apache Ant tool.

For more information about building a Java RSSE application using Apache Ant, see "How to build a Java RSSE sample application," earlier in this chapter.

4 Copy the file, rsseAcl.jar, to the lib directory of the BIRT iHub servlet container and configure the class.properties file.

For more information about copying the archive file for a Java RSSE application to the lib directory for the BIRT iHub servlet container and configuring the class.properties file, see "How to install a Java RSSE application on an Encyclopedia volume," earlier in this chapter.

5 Configure the Encyclopedia volume to use open security as a web service.

For more information about enabling an RSSE application to run as a web service on an Encyclopedia volume, see "How to enable the open security web service to use the Java RSSE application," earlier in this chapter.

Creating an access control list (ACL)

The file, user.acls, stores a user's access control list (ACL) using the following format:

```
Username=acl1, acl2, ..
```

The user name field matches the name of user in the Encyclopedia volume. An equal ('=') sign separates the user name from the ACL list. An ACL list can contain zero, one, or more ACL specifications, as shown in the following code example:

```
user1=acl1, acl2, acl3, acl4
user2=acl5, acl6, acl7, acl8
user3=acl9
user4=acl10
```

If there is more than one ACL specification in the list, separate each ACL using a comma. The scanner reading the users.acls file eliminates any white space or backslash.

All the user name specifications in the example are legal. A list can contain users that do not appear in the Encyclopedia. The information for these users is ignored.

Deploying a report to an Encyclopedia volume

Test page-level security by deploying the sample design to the Encyclopedia volume. For example, a report may shows information about the sales reps in the following city offices:

- NYC
- Boston
- Philadelphia

User1 has access to the pages with information about NYC office, user2 to the Boston office, and user3 to the Philadelphia office. The file, user.acls, contains the following access control list specifications:

```
user1=NYC
user2=Boston
user3=Philadelphia
```

To deploy the sample design to an Encyclopedia volume, perform the following steps:

1. Using iHub Management Console, log in to the Encyclopedia volume as Administrator.
2. In Files and Folders, choose Add File and upload the design to the Encyclopedia volume.
3. In Files and Folders, from the design file drop-down list, choose Run to execute the design.

   On Parameters, select Save the output document to create a document on the Encyclopedia volume.

   Choose OK to view the document output.

   The administrator is able to see all three offices.
4. On Users, choose Create User to create a new user.

   In New User, create user1, as shown in Figure 10-4.

   Choose OK.

   Repeat step 4 to create user2 and user3.

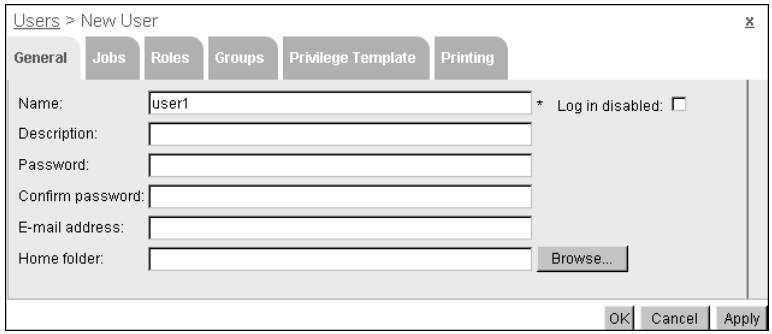


Figure 10-4 Creating a new user

5 In Files and Folders, from the design file drop-down list, choose Properties, then perform the following operations:

1. To set the privileges on the design file, choose Privileges.
2. On Privileges, in Available, select All, then choose the right arrow to copy All to Selected.
3. Select Read.

   Privileges for All on office_rep.rptdesign looks like Figure 10-5.

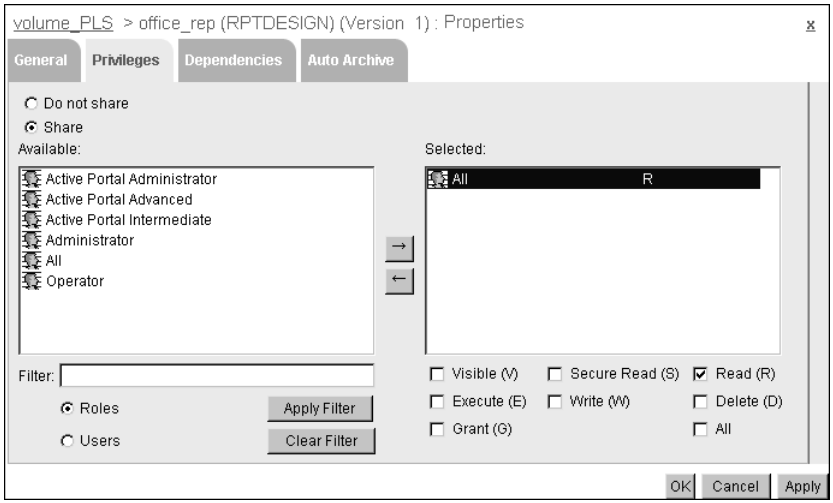


Figure 10-5 Specifying user privileges

4. Choose OK.

6 On Files and Folders, from the document drop-down list, choose Properties, then perform the following operations:

1. To set the privileges on the design file, choose Privileges.

2. On Privileges, in Available, select All, then choose the right arrow to copy All to Selected.
3. Select Visible and Secure Read.

7. Log out from the Encyclopedia volume.
8. Log in to the Encyclopedia volume as user1.
9. View the document.

   User1 can only see the information for the NYC office, as shown in Figure 10-6.

Figure 10-6 Example of document output

10. Repeat steps 8 and 9, logging in as user2, then user3.

    User2 can only see the information for the Boston office and user3 can only see the information for the Philadelphia office.

To change the assignments in the file, user.acls, wait for the volume cache time-out period. Alternatively, put the Encyclopedia volume offline, restart the BIRT iHub application container, then take the Encyclopedia volume online again before checking to see if the changes are effective.

SOAP-based Report Server Security Extension (RSSE) operations

This section describes the SOAP-based RSSE operations.

Authenticate

Verifies that the user is authorized to access the BIRT iHub System. Implement Authenticate for external user authentication and external user registration.

Request schema

```
<xsd:complexType name="Authenticate">
   <xsd:sequence>
      <xsd:element name="User" type="xsd:string"/>
      <xsd:element name="Password" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="Credentials" type="xsd:base64Binary"
         minOccurs="0"/>
      <xsd:element name="UserSetting" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

User
The name of the user logging on to BIRT iHub.

Password
The user's password.

Credentials
Additional credentials for authenticating the user.

UserSetting
Specifies whether to return the user's properties. If True, returns the user's properties.

Response schema

```
<xsd:complexType name="AuthenticateResponse">
   <xsd:sequence>
      <xsd:element name="UserAndProperties"
         type="typens:UserAndProperties" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

UserAndProperties
The user's name and properties.

DoesGroupExist

Verifies whether the group exists in the external directory. BIRT iHub can call this function to clear references to deleted groups.

Request schema

```
<xsd:complexType name="DoesGroupExist">
   <xsd:sequence>
      <xsd:element name="GroupName" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

GroupName
The name of the group to verify.

Response schema

```
<xsd:complexType name="DoesGroupExistResponse">
   <xsd:sequence>
      <xsd:element name="Exists" type="xsd:boolean"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Exists
Indicates whether the group exists. If True, the group exists.

DoesRoleExist

Verifies whether the role exists in the external directory. BIRT iHub can call this function to clear references to deleted roles.

Request schema

```
<xsd:complexType name="DoesRoleExist">
   <xsd:sequence>
      <xsd:element name="RoleName" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

RoleName
The name of the role to verify.

Response schema

```
<xsd:complexType name="DoesRoleExistResponse">
   <xsd:sequence>
      <xsd:element name="Exists" type="xsd:boolean"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Exists
Indicates whether the role exists. If True, the role exists.

DoesUserExist

Verifies whether the user exists in the external directory. BIRT iHub can call this function to clear references to deleted users.

Request schema

```
<xsd:complexType name="DoesUserExist">
   <xsd:sequence>
      <xsd:element name="UserName" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UserName
The name of the user to verify.

Response schema

```
<xsd:complexType name="DoesUserExistResponse">
   <xsd:sequence>
      <xsd:element name="Exists" type="xsd:boolean"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Exists
Indicates whether the user exists. If True, the user exists.

GetConnectionProperties

Retrieves the connection properties for a user or role from an external data source for a pass-through security operation. In pass-through security, an information object's DCD file sets the securityPolicy to TranslatedCredential. The proxy user name and password settings, specifying the user login credentials in the DCD, contain empty quotes and are ignored by the implementation.

Request schema

```
<xsd:complexType name="GetConnectionProperties">
   <xsd:sequence>
      <xsd:element name="FileName" type="xsd:string"/>
      <xsd:element name="UserName" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

FileName
The fully qualified name of an information object's data connection definition (DCD) file.

UserName
The name of the user or role.

Response schema

```
<xsd:complexType name="GetConnectionPropertiesResponse">
   <xsd:sequence>
      <xsd:element name="ConnectionProperties"
         type="typens:ArrayOfPropertyValue"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ConnectionProperties
The requested name and value pairs.

GetTranslatedRoleNames

Maps the external security role names to Actuate security role names. Either use GetTranslatedRoleNames in conjunction with the external registration security level, or use the same role names for the external and Actuate roles.

For example, a user with the Actuate Administrator security role can manage all items in an Encyclopedia volume. If the Administrator role in the external security system has a different meaning, GetTranslatedRoleNames can map the external security role to an Actuate role with a different name.

Request schema

```
<xsd:complexType name="GetTranslatedRoleNames"/>
   <xsd:element name="GetTranslatedRoleNames"
      type="typens:GetTranslatedRoleNames" />
```

Response schema

```
<xsd:complexType name="GetTranslatedRoleNamesResponse">
   <xsd:sequence>
      <xsd:element name="TranslatedRoleNames"
         type="typens:TranslatedRoleNames"/>
</xsd:complexType>
```

Response elements

TranslatedRoleNames
The names that Actuate uses for external security roles.

GetTranslatedUserNames

Maps the external security user names to Actuate security user names. Either use GetTranslatedUserNames in conjunction with the external registration security level, or use the same user names for the external and Actuate users.

For example, a user with the Actuate Administrator privilege can manage all items in an Encyclopedia volume. If the Administrator user in the external security system has a different meaning, GetTranslatedUserNames can map the external security user to an Actuate user with a different name.

Request schema

```
<xsd:complexType name="GetTranslatedUserNames"/>
   <xsd:element name="GetTranslatedUserNames"
      type="typens:GetTranslatedUserNames" />
```

Response schema

```
<xsd:complexType name="GetTranslatedUserNamesResponse">
   <xsd:sequence>
      <xsd:element name="TranslatedUserNames"
         type="typens:TranslatedUserNames"/>
</xsd:complexType>
```

Response elements

TranslatedUserNames
The names that Actuate uses for external security users.

GetUserACL

Retrieves the user's ACL. GetUserACL applies only if using page-level security. Page-level security controls printing, navigating, and all aspects of user viewing. Page-level security requires the Page Level Security option on BIRT iHub.

Request schema

```
<xsd:complexType name="GetUserACL">
   <xsd:sequence>
      <xsd:element name="UserName" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

UserName
The name of the user whose ACL to retrieve.

Response schema

```
<xsd:complexType name="GetUserACLResponse">
   <xsd:sequence>
      <xsd:element name="ACL" type="typens:ArrayOfString"/>
   </xsd:sequence>
</xsd:complexType>
</xsd:element>
```

Response elements

ACL
The list of pages of a document to which the user has access.

GetUserProperties

Retrieves the user's properties from an external directory. Regardless of security level implementation, implement GetUserProperties when the user's properties are stored in an external security source.

Request schema

```
<xsd:complexType name="GetUserProperties">
   <xsd:sequence>
      <xsd:element name="Users" type="typens:ArrayOfString"/>
      <xsd:element name="ResultDef" type="typens:ArrayOfString">
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

User
The name of the user whose properties to retrieve.

ResultDef
The properties to retrieve.

Response schema

```
<xsd:complexType name="GetUserPropertiesResponse">
   <xsd:sequence>
      <xsd:element name="ArrayOfUserAndProperties"
         type="typens:ArrayOfUserAndProperties"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ArrayOfUserAndProperties
The user properties.

GetUsersToNotify

Retrieves the list of users to notify about completed jobs.

Request schema

```
<xsd:complexType="GetUsersToNotify"/>
```

Response schema

```
<xsd:complexType name="GetUsersToNotifyResponse">
   <xsd:sequence>
      <xsd:element name="Users" type="typens:ArrayOfString"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Users
The list of users to notify.

PassThrough

Calls the RSSE for general purposes such as changing or refreshing the internal library state. If implemented, the RSSE calls PassThrough in response to the BIRT iHub receiving the Information Delivery API CallOpenSecurityLibrary request.

The RSSE passes the ReturnCode as a response to CallOpenSecurityLibrary, RSSE does not interpret the parameter.

Request schema

```
<xsd:complexType name="PassThrough">
   <xsd:sequence>
      <xsd:element name="Input" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

Input
The input parameter string.

Response schema

```
<xsd:complexType name="PassThroughResponse">
   <xsd:sequence>
      <xsd:element name="Output" type="xsd:string"/>
      <xsd:element name="ReturnCode" type="xsd:int"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Output
The output parameter string.

ReturnCode
The integer parameter that the caller of CallOpenSecurityLibrary interprets.

SelectGroups

Retrieves the names of groups that match the specified criteria. To retrieve a list of a user's group memberships, specify a name in UserName. SelectGroupsResponse then returns the list of the user's groups. SelectGroups is required when using external registration security level.

Request schema

```
<xsd:complexType="SelectGroups">
   <xsd:sequence>
      <xsd:sequence>
         <xsd:element name="QueryPattern" type="xsd:string"/>
         <xsd:element name="UserName" type="xsd:string"/>
      </xsd:sequence>
      <xsd:element name="FetchSize" type="xsd:int" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

QueryPattern
The string match.

UserName
The name of a user whose group membership to retrieve. Must not be an empty string.

FetchSize
The maximum number of records to retrieve and return in a result set. The default value is 500.

Response schema

```
<xsd:element name="SelectGroupsResponse">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="Groups" type="typens:ArrayOfString"/>
         <xsd:element name="TotalCount" type="xsd:int"/>
      </xsd:sequence>
```

```
   </xsd:complexType>
</xsd:element>
```

Response elements

Groups
The list of users matching the search criteria.

TotalCount
The number of entries in the search result set.

SelectRoles

Searches for roles that match the specified criteria. Required if using an external registration security level.

SelectRoles can also retrieve a user's roles. To retrieve a user's roles, specify a name in UserName. SelectRolesResponse then returns the list of the user's roles.

The SelectRoles SOAP message invokes the SelectRolesOfUser method within the Java code, and does not invoke the SelectRoles method. The Security Roles tab in the Management Console invokes the SelectRoles method. iHub does not use the SelectRoles method to link a user account to a role.

Request schema

```
<xsd:complexType name="SelectRoles">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="QueryPattern" type="xsd:string"/>
         <xsd:element name="UserName" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="FetchSize" type="xsd:int" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

QueryPattern
The string match.

UserName
The name of a user whose information to retrieve.

FetchSize
The maximum number of records to retrieve and return in a result set. The default value is 500.

Response schema

```
<xsd:complexType name="SelectRolesResponse">
   <xsd:sequence>
      <xsd:element name="Roles" type="typens:ArrayOfString"/>
      <xsd:element name="TotalCount" type="xsd:int"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

Roles
The list of roles matching the search criteria.

TotalCount
The number of entries in the search result set.

SelectUsers

Retrieves the names of users that match the specified criteria. For example, to retrieve the names of all users in the Sales group, specify Sales in GroupName.

SelectUsers is required if using an external registration security level.

Request schema

```
<xsd:complexType name="SelectUsers">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="QueryPattern" type="xsd:string"/>
      <xsd:element name="RoleName" type="xsd:string"/>
         <xsd:element name="GroupName" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="FetchSize" type="xsd:int" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

QueryPattern
The string match.

RoleName
The name of the role whose members to retrieve.

GroupName
The name of the group whose members to retrieve.

FetchSize
The maximum number of records to retrieve and return in a result set. The default value is 500.

Response schema

```
<xsd:complexType name="SelectUsersResponse">
      <xsd:sequence>
         <xsd:element name="Users" type="typens:ArrayOfString"/>
         <xsd:element name="TotalCount" type="xsd:int"/>
      </xsd:sequence>
</xsd:complexType>
```

Response elements

Users
The list of users matching the search criteria.

TotalCount
The number of entries in the search result set.

Start

Initializes the RSSE. Implement Start to initialize RSSE.

Request schema

```
<xsd:complexType name="Start">
   <xsd:sequence>
      <xsd:element name="ServerHome" type="xsd:string"/>
      <xsd:element name="Volume" type="xsd:string"/>
      <xsd:element name="LogFile" type="xsd:string"/>
      <xsd:element name="Version" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

ServerHome
The path to the BIRT iHub installation, for example C:\Program Files\Actuate\Server on Windows.

Volume
The name of the Encyclopedia volume.

LogFile
The path to the log file for RSSE activity.

Version
The BIRT iHub version number.

Response schema

```
<xsd:complexType name="StartResponse">
   <xsd:sequence>
      <xsd:element name="IntegrationLevel" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="ExternalProperties"
         type="typens:ArrayOfString" minOccurs="0"/>
      <xsd:element name="RSSEVersion" type="xsd:string"/>
      <xsd:element name="UserACLExternal" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="ConnectionPropertyExternal"
         type="xsd:boolean" minOccurs="0"/>
      <xsd:element name="SelectUsersOfRole" type="xsd:boolean"
         minOccurs="0"/>
      <xsd:element name="SelectGroupsOfUser" type="xsd:boolean"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

IntegrationLevel
The integration level of external security. One of the following values:

- External_Authentication

- External_Registration
- None

ExternalProperties
One or more of the following external user or role properties:

- EmailAddress
  The user's e-mail address
- HomeFolder
  The user's home folder
- EmailForm
  The form of the e-mail attachment, included or linked
- EmailWhen
  The type of notification to use for a completed job
- FolderWhen
  An indicator of when the user uses the Completed folder
- SuccessNoticeExpiration
  The number of minutes that elapse before the Encyclopedia service deletes a successful job notice
- FailNoticeExpiration
  The number of minutes that elapse before the Encyclopedia service deletes a failed job notice
- DefaultObjectPrivileges
  The privileges that the user has by default on the objects the user creates
- MaxPriority
  The maximum request priority the user can set when creating a report printing or generation request
- ViewPreference
  The user's web viewing preference, default or DHTML
- ChannelSubscriptionList
  A list of channels to which the user subscribes

RSSEVersion
The version of RSSE.

UserACLExternal
Specifies whether the user's access list is stored externally. Applies only if using page-level security.

ConnectionPropertyExternal
Specifies whether the user's connection properties are retrieved externally from the RSSE. If True, the connection properties are retrieved externally. In this case, BIRT iHub directs requests to set connection properties to the RSSE and does not use GetConnectionProperties.

SelectUsersOfRole
Applies only under external registration. Specifies whether the Role element in SelectUsers is implemented. The setting indicates whether BIRT iHub enables this feature. The default value is False.

SelectGroupsOfUser
Applies only under external registration. Specifies whether the User element in SelectGroups is implemented. The setting indicates whether BIRT iHub enables this feature. The default value is False.

Stop

Stops the RSSE. Implement Stop to close the RSSE and free system resources.

Request schema

```
<xsd:complexType name="Stop"/>
```

Response schema

```
<xsd:complexType name="StopResponse"/>
```

SOAP-based Report Server Security Extension (RSSE) data types

This section describes the SOAP-based RSSE data types. Some data types have the same name as data types within the IDAPI, but do not have the same content.

Arrays of data types

Data type definitions can be grouped into arrays. Each array has a specific definition for that data type.

The schema for an array of a data type generally follows the following pattern:

```
<xsd:complexType name="ArrayOfX">
   <xsd:sequence>
      <xsd:element name="X" type="typens:X"
         maxOccurs="unbounded"/>
```

```
   </xsd:sequence>
</xsd:complexType>
```

In the above listing, X is the data type of object the array contains. For example, the XML for an array of Aggregation objects is:

```
<xsd:complexType name="ArrayOfPropertyValue">
   <xsd:sequence>
      <xsd:element name="PropertyValue"
         type="typens:PropertyValue"
         maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

The following data types have arrays defined and used in the RSSE:

- Permission
- PropertyValue
- String
- UserAndProperties

Permission

A complex data type that describes a user's access rights.

Schema

```
<xsd:complexType name="Permission">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="RoleName" type="xsd:string" />
         <xsd:element name="UserName" type="xsd:string" />
      </xsd:choice>
      <xsd:element name="AccessRight" type="xsd:string" />
   </xsd:sequence>
  </xsd:complexType>
```

Elements

RoleName
The role name.

UserName
The user name.

AccessRight
The privileges the user or role has on an object. One or more of the following characters representing a privilege:

- D—Delete
- E—Execute

- G—Grant
- V—Visible
- S—Secured Read
- R—Read
- W—Write

PropertyValue

A complex data type that describes a name-value pair.

Schema

```
<xsd:complexType name="PropertyValue">
   <xsd:all>
      <xsd:element name="Name" type="xsd:string" />
      <xsd:element name="Value" type="xsd:string" minOccurs="0" />
   </xsd:all>
</xsd:complexType>
```

Elements

Name
The name portion of the name-value pair.

Value
The value portion of the name-value pair.

TranslatedRoleNames

A complex data type that describes the role names RSSE uses that match external role names.

Schema

```
<xsd:complexType name="TranslatedRoleNames">
   <xsd:all>
      <xsd:element name="Administrator" type="xsd:string" />
      <xsd:element name="Operator" type="xsd:string" />
      <xsd:element name="All" type="xsd:string" />
   </xsd:all>
  </xsd:complexType>
```

Elements

Administrator
The Administrator role name.

Operator
The Operator role name.

All
All other role names.

User

A complex data type describing an RSSE user and their attributes.

Schema

```
<xsd:complexType name="User">
   <xsd:all>
      <xsd:element name="Name" type="xsd:string" />
      <xsd:element name="EmailAddress" type="xsd:string"
         minOccurs="0" />
      <xsd:element name="HomeFolder" type="xsd:string"
         minOccurs="0" />
      <xsd:element name="LicenseOptions"
         type="typens:ArrayOfString" minOccurs="0" />
      <xsd:element name="ViewPreference" type="xsd:string"
         minOccurs="0" />
      <xsd:element name="MaxJobPriority" type="xsd:long"
         minOccurs="0" />
      <xsd:element name="SendNoticeForSuccess" type="xsd:boolean"
         minOccurs="0" />
      <xsd:element name="SendNoticeForFailure" type="xsd:boolean"
         minOccurs="0" />
      <xsd:element name="SuccessNoticeExpiration" type="xsd:long"
         minOccurs="0" />
      <xsd:element name="FailureNoticeExpiration" type="xsd:long"
         minOccurs="0" />
      <xsd:element name="SendEmailForSuccess" type="xsd:boolean"
         minOccurs="0" />
      <xsd:element name="SendEmailForFailure" type="xsd:boolean"
         minOccurs="0" />
      <xsd:element name="AttachReportInEmail" type="xsd:boolean"
         minOccurs="0" />
      </xsd:all>
</xsd:complexType>
```

Elements

Name
The user's name.

EmailAddress
The user's e-mail address.

HomeFolder
The users's home folder.

LicenseOptions
The user's license options.

ViewPreference
The user's viewer, Default or DHTML.

MaxJobPriority
The maximum priority that the user can assign to a job.

SendNoticeForSuccess
Specifies whether the BIRT iHub sends success notices to the user.

SendNoticeForFailure
Specifies whether the BIRT iHub sends failure notices to the user.

SuccessNoticeExpiration
Specifies the minimum number of minutes success notices remain in the Completed folder. To set the user's success notices to never expire, set the value to 0xffffffff.

FailureNoticeExpiration
Specifies the minimum number of minutes failure notices remain in the Completed folder. To set the user's failure notices to never expire, set the value to 0xffffffff.

SendEmailForSuccess
Specifies whether the BIRT iHub sends success notices in an e-mail message to the user.

SendEmailForFailure
Specifies whether the BIRT iHub sends failure notices in an e-mail message to the user.

AttachReportInEmail
Specifies whether to attach a report to an e-mail completion notice.

Chapter

11

Using Actuate logging and monitoring APIs

This chapter contains the following topics:

- About usage logging and error logging extensions
- Installing and using usage and error logging extensions
- Customizing the usage logging extension
- Customizing the error logging extension
- About the usage log
- About the error log
- About BIRT iHub usage and error log consolidator

About usage logging and error logging extensions

BIRT iHub System provides a monitoring framework that logs BIRT iHub usage and error information. You can use this information to understand how BIRT iHub uses system resources and troubleshoot problems.

BIRT iHub and iHub Integration Technology provide usage and error logging extensions that retrieve and write log data to files. The usage and error logging extensions are DLLs on a Windows platform and shared libraries on a UNIX system. BIRT iHub Integration Technology provides the customizable source code for the usage and error logging extensions as reference implementations.

BIRT iHub Integration Technology provides a reference implementation of an usage and error log consolidator application that reads data from usage and error log files and inserts the information in a database. Actuate also provides an extension to the Windows system monitoring tool that collects data on BIRT iHub System resources.

Installing and using usage and error logging extensions

The usage and error logging extensions are open framework applications. These reference implementations log the information that the BIRT iHub monitoring framework captures to files. A developer can customize the way the DLL or shared library handles the usage and error log information. For more information about these usage and error log extensions, refer to the readme files in ACTUATE_HOME/ServerIntTech/User Activity Logging Extension and Error Logging Extension.

BIRT iHub installs compiled versions of the usage and error logging extensions in AC_SERVER_HOME/bin. In Windows, the DLL applications files are UsrActivityLoggingExt.dll and ErrorLoggingExt.dll. In Linux and UNIX, the shared library applications files are libUsrActivityLoggingExt.so and libErrorActivityLoggingExt.so.

A usage or error log file is a comma-separated values (CSV) file. The default name for a usage log file is usage_log_1.csv. The default name for an error log file is error_log_1.csv.

In Actuate Release 10, the usage and error logs for each volume were written to separate directories. In Release 11 and iHub, the usage and error logs for all volumes are consolidated in one directory, AC_SERVER_HOME/<version> /UsageErrorLogs/primary, and written to one usage and one error log. The directory for the usage and error log files is not configurable.

How to configure usage logging

To configure usage logging, perform the following tasks:

1. Log in to iHub Configuration Console and choose Advanced View.
2. On System—Status, choose Properties.
3. On System—Properties—General, choose Usage Logging.

   System—Properties—Usage Logging appears as shown in Figure 11-1.

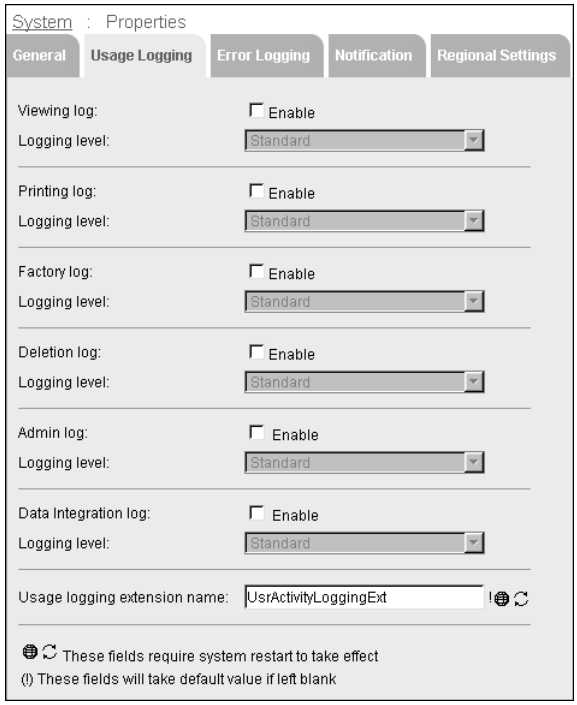


Figure 11-1 Configuring usage logging

4. Select the usage logging information you want to capture from the following list of logging options:
   - Viewing
   - Printing
   - Factory
   - Deletion
   - Admin
   - Data Integration

1. Select Enable to activate the logging option.
2. Select Standard or Detail for the logging level.

   For viewing, deletion, and printing logging, standard and detail information are the same in the logging application that ships with BIRT iHub.

   For Factory logging, detailed information includes report parameters. Logging detailed Factory information, instead of standard Factory information, causes performance degradation.

5. In Usage logging extension name, enter the name of the usage logging extension.

   UsrActivityLoggingExt is the name of the default usage logging extension.

   Choose OK.

How to configure error logging

To configure usage logging, perform the following tasks:

1. Log in to iHub Configuration Console and choose Advanced View.
2. On System—Status, choose Properties.
3. On System—Properties—General, choose Error Logging.

   System—Properties—Error Logging appears as shown in Figure 11-2.

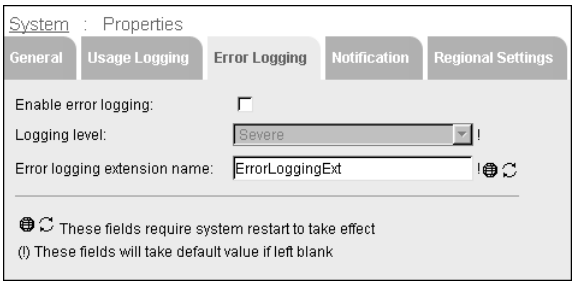


Figure 11-2 Configuring error logging

4. Select Enable error logging.
5. Select the error logging level you want to capture from the following list of options:

   - Information
     Logs informational messages to helps track BIRT iHub behavior.
   - Warning
     Logs warning errors that typically do not impact normal BIRT iHub operation.

- Severe

  Logs errors that can cause BIRT iHub to abort execution if you do not rectify the cause of the error. A severe error does not typically cause BIRT iHub to abort execution immediately.

- Fatal

  Logs critical errors from which BIRT iHub cannot recover. A fatal error typically causes BIRT iHub to abort execution immediately.

6 In Error logging extension name, enter the name of the error logging extension.

  ErrorLoggingExt is the name of the default error logging extension.

  Choose OK.

Customizing the usage logging extension

To customize the usage logging extension, you must install BIRT iHub Integration Technology. BIRT iHub Integration Technology provides the C and C++ source code that you modify to customize the usage logging extension. The usage logging extension source code file, usagelogext.c, installs in $ACTUATE_HOME/ServerIntTech/User Activity Logging Extension.

The usagelogext.c source code implements the following functionality:

- Specifies the name and extension of the usage log file, and the location or path to the file

  These properties are defined as constants. For example:

  ```
  #define USAGELOG_FILE_NAME "usage_log"
  #define USAGELOG_FILE_EXT ".csv"
  ```

  The AcStartUsageLog function implements this functionality.

- Writes information about every transaction that BIRT iHub captures to a log file

  The AcLogUsage function implements this functionality.

- Stops logging information and releases the resources the extension uses

  The AcStopUsageLog function implements this functionality.

- Specifies whether the extension is multithread-safe

  The AcIsThreadSafe function implements this functionality. If the extension is not multithread-safe, AcIsThreadSafe must return False. The reference implementation is not multithread-safe.

Customizing the error logging extension

To customize the Error Logging extension, you must install BIRT iHub Integration Technology. BIRT iHub Integration Technology provides the C and C++ source code that you modify to customize the Error Logging extension. The Error Logging extension source code file, usagelogext.c, installs in $ACTUATE_HOME/ServerIntTech/Error Logging Extension.

The errorlogext.c source code implements the following functionality:

- Specifies the name and extension of the log file, and the location or path to the log file

  These properties are defined as constants. For example:

  ```
  #define ERRORLOG_FILENAME "error_log"
  #define ERRORLOG_FILE_EXT ".csv"
  ```

  The AcStartErrorLog function implements this functionality.

- Writes the information about every error that BIRT iHub encounters to a log file

  The AcLogError function implements this functionality.

- Stops logging information and releases the resources the extension uses

  The AcStopErrorLog function implements this functionality.

- Specifies whether or not the extension is multithread-safe

  The AcIsThreadSafe function implements this functionality. If the extension is not multithread-safe, AcIsThreadSafe must return False. The reference implementation is not multithread-safe.

About the usage log

The usage log, usage_log.csv, is a comma-separated values (CSV) file. The usage log records the following events:

- Report viewing
- Report printing
- Report generation
- Report deletion
- Administrative
- Data integration

About types of recorded events

For each type of event, you can set the logging level to Standard or Detail. If you are using the default usage logging extension, UsrActivityLoggingExt, the logging level does not affect how the file records the following types of events:

- Report viewing
- Report printing
- Report deletion

If you set the logging level for report generation or factory events to Detail, the usage log includes report parameters. Setting the logging level to Detail for report generation events decreases performance.

Understanding a usage log entry

Each usage log entry is a comma-separated list containing up to 40 fields of information about an event. The following example describes a delete user event:

```
3272649170,5,1,3272649170,3272649170,-,-,0,Administrator,3,
   enl2509,enl2509,enl2509,User,testUser,-,-,-,-,-,-,-,-,-,-,
   2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

The usage log organizes the entry fields into the following information groups:

- Fields 1 through 10 contain general information:
  - Fields 1, 4, and 5 contain the log file time stamp, start time, and finish time. The time is in seconds since 00:00:00, Jan. 1, 1901, GMT.
  - Field 2 contains the event type. The numeric values in Table 11-1 indicate the event types.

Table 11-1 Event types and the corresponding event values

Event type	Event value
ReportGeneration	1
ReportPrinting	2
ReportViewing	3
ReportDeletion	4
Admin	5
Query	6
Search	7

- Field 3 contains the event result. The value for the event result is either 1 or 0, indicating success or failure.
- Fields 6 through 8 contain report output information, indicating the file name, version, and file size. The report output group information appears only with report events. A dash indicates a field is not used.
- Fields 9 and 10 contain execution information, indicating the user name and the BIRT iHub subsystem where the operation executed. The numeric values in Table 11-2 indicate the BIRT iHub subsystems.

Table 11-2 BIRT iHub subsystems and the corresponding ID numbers

Subsystem	ID number
ReportEngine	1
ViewEngine	2
EncycEngine	3
IntegrationEngine	4
Cache	5

- Fields 11 through 25 contain operational information in string format, including the Encyclopedia volume, BIRT iHub, and cluster names. Fields 26 through 40 contain operational information in numeric format.

  The values in these fields depend on the value for the event type in field 2. Table 11-3 summarizes some of the information available for each event type at Standard level.

Table 11-3 Examples of information that is available about the different types of events

Event type	Event value	Operation data available
Report generation	1	String fields 11 through 21 display the following information: – `,executable name, executable version, volume name, server name, cluster name, resource group name, node running request, page count, job name, request ID` A dash indicates a field is not used. Numeric fields 26 through 29 display the following information: `number of pages,submit time, job type, job priority`

Table 11-3 Examples of information that is available about the different types of events

Event type	Event value	Operation data available

Report printing	2	String fields 11 through 18 display the following information:
		`page numbers printed, volume name, printer name, server name, cluster name, node sent to, file type, server request id`
		Numeric fields 26 through 29 display the following information:
		`number of pages printed, submit time, job type, job priority`

Report viewing	3	String fields 11 through 18 display the following information: `output format, report page numbers, volume name, server name, cluster name` Numeric field 26 displays the number of pages viewed.
Administrative	5	String fields 11 through 13 display the following information: `volume name, server name, cluster name` Numeric field 26 displays an operation ID for an administration event. The following list provides the event name for each operation ID: ■ 1 Create ■ 2 Delete ■ 3 Modify ■ 4 Login

Actuate Integration service	6	String fields 11 through 14 display the following information:
		`volume name, server name, cluster name, server request id`
		Numeric fields 26 and 27 display the following information:
		`request wait time, request generation time`

Search	7	String fields 11 through 15 display the following information: `report format, page numbers, volume name, server name, cluster name` Numeric field 26 displays the number of pages viewed.

About the error log

The error log, error_log.csv, is a comma-separated values (CSV) file. If you use the default error logging extension, ErrorLoggingExt, you can set the logging level to:

- Information
  The error log records messages that trace BIRT iHub behavior.
- Warning
  The error log records warnings. The errors do not necessarily affect the operation of BIRT iHub.
- Severe
  The error log records errors that can result in BIRT iHub failure if you do not correct them.
- Fatal
  The error log records critical errors from which BIRT iHub cannot recover and that can result in failure.

Understanding an error log entry

Each error log entry is a comma-separated list containing up to 12 fields about an error-related event. The following example describes an error in a submit job event:

```
3272648796,2,3230,SubmitJob,Administrator,"Invalid start time or
   end time.",enl2509,enl2509,enl2509,-,-,-
```

The error log organizes the entry fields into the following information groups:

- Fields 1 through 9 contain general information:
  - Field 1 contains the log file time stamp. The time is in seconds since 00:00:00, Jan. 1, 1901, GMT.
  - Field 2 contains the error severity level, an integer between 1 and 4. The numeric values in Table 11-4 indicate the level.
  - Field 3 contains the Error ID code.

Table 11-4 Error severity levels and the corresponding values

Error severity level	Value
Information	1
Warning	2
Severe	3
Fatal	4

  - Field 4 contains the Service name, indicating the subsystem where the error occurred such as the Factory, Encyclopedia, View, or Request service.
  - Field 5 indicates the Encyclopedia volume user.

- Field 6 contains the error message.
- Field 7 contains the Encyclopedia volume name.
- Field 8 contains the BIRT iHub cluster name.
- Field 9 contains the BIRT iHub node name.
- Depending on the error, fields 10 through 12 can contain information such as a file name and ID number. A dash indicates a field is not used.

  Table 11-5 summarizes some of the information available in fields 10 through 12 for an error log entry at Standard level.

Table 11-5 Information that is available for error log entries at the Standard level

Type of error	Operation data available
Cluster master failover	Fields 10 and 11 display the following data: ■ Original cluster master ■ New cluster master
Encyclopedia volume user activity	Fields 10 through 12 can contain error parameters such as the following items: ■ Object name ■ ID number
Volume failover	Fields 10 and 11 contain the following data: ■ Primary server ■ Backup server used
Volume online or offline	Fields 10 and 11 contain the following data: ■ Volume name ■ Operation type either online or offline
BIRT iHub node start or stop	Field 10 contains the BIRT iHub name
Service enable or disable	Fields 10 and 11 contain the following data: ■ Server name ■ List of services
Archive service error	Fields 10 through 12 contain error parameters
Encyclopedia volume job purging field 4 is Job Purge	Fields 10 through 12 contain error parameters
Encyclopedia volume health monitoring field 4 is Encyclopedia Health Monitor	Fields 10 through 12 contain error parameters

About BIRT iHub error messages

Table 11-6 lists the general categories of BIRT iHub error messages.

Table 11-6 Categories of BIRT iHub error messages

Error ID range	Error description
0001 - 1000	System errors such as Out of memory or Low thread count
1001 - 3000	BIRT iHub errors such as Corrupt encyclopedia or Transient storage full Within this error category, the following sub-categories exist: ■ 1001 - 2000 Actuate internal datastore ■ 2001 - 3000 Actuate internal
3001 - 6000	User errors such as Permission denied Within this error category, the following sub-categories exist: ■ 3001 - 4000 Encyclopedia engine ■ 4001 - 5000 Report engine ■ 5001 - 6000 View engine
6001 - 12000	■ 6001 - 7000 SOAP engine ■ 7001 - 8000 Process management daemon ■ 8001 - 9000 Cluster engine ■ 10001 - 11000 Server configuration ■ 11001 - 12000 XML parsing
12001 - 13000	Viewing server errors
13000 - 14000	AcMail exceptions
100001 - 100600	Actuate Information service
100601 - 100699	Actuate Caching service
100700 - 150000	Shared by Actuate Information service and Actuate Caching service

About BIRT iHub usage and error log consolidator

The log consolidator application is a Java application that reads data from an BIRT iHub usage or error log file and uses JDBC to add the information to a database. In an BIRT iHub cluster, you must install and run the log consolidator application on each BIRT iHub node to consolidate the cluster's usage and error

log information in a database. Before running the log consolidator application, you must install the following components:

- BIRT iHub
- Log consolidator application files
- Log consolidator configuration file
- Database used by the log consolidator application

BIRT iHub installs the required JAR files for the log consolidator application in $ACTUATE_HOME/Jar/UsageAndErrorConsolidator. These files include:

- usageanderrorconsolidator.jar
  The com.actuate.consolidator application class and properties files.
- Java Architecture for XML Binding (JAXB) JAR files
  JAXB provides a framework that supports run-time mapping between XML and Java objects.
- ojdbc14.jar
  The supported Oracle JDBC Driver. The reference implementation uses Oracle as the example database.
- naming-java.jar
  Contains the handler for the Java namespace.

The log consolidator application also uses the following Microsoft Windows Registry key or UNIX environment variable set by BIRT iHub installation process:

- On Windows, make sure the following registry key exists:

  ```
  HKEY_LOCAL_MACHINE\SOFTWARE\Actuate\Common\9.0\AC_JRE_HOME
  ```

  The log consolidator application uses java.exe from:

  ```
  AC_JRE_HOME\bin
  ```

- On UNIX, make sure the following environment variable exists:

  ```
  AC_JRE_HOME
  ```

  The log consolidator application uses java.exe from:

  ```
  $AC_JRE_HOME/bin
  ```

In BIRT iHub Integration Technology, the UsageAndErrorConsolidator directory contains additional files that you must use to complete the installation of the log consolidator application:

- /DBScripts contains the SQL script, CreateActuateLogTables.sql, which creates the tables in the Oracle database used by the Actuate log consolidator sample application. A readme.txt file describes this SQL script file.

- /jar contains the JAR files required by the log consolidator application.
- /Setup contains the following files that startup and shutdown the log consolidator application:
  - consolidatorconfig.xml is the sample consolidator configuration file that specifies settings such as the following items:
    - Database driver, URL, encoding, schema, user name, and password
    - Usage and error log details, such as the file names, refresh interval, number of logs, and whether a log file is enabled
  - The UNIX version uses the scripts, start_consolidator.sh and stop_consolidator.sh, to start up and shut down the log consolidator application.
  - The Windows version uses a setup application, consolidatorwin.exe, that installs the log consolidator application as a Windows service and starts and stops the application.
  - A readme.txt file describes how to use these components.
- /src contains the following items:
  - usageanderrorconsolidator.jar, the JAR file for the log consolidator application
  - consolidatormake.xml, an Ant build file
  - com.actuate.consolidator, the log application source code

How to install the log consolidator application

1. Edit the following settings in consolidatorconfig.xml:
   - JDBC driver name
   - URL, specifying the type of JDBC driver and database connection information, including host name, port, and database instance (SID) or service name, using the following syntax:

     ```
     jdbc:oracle:thin:@//[HOST][:PORT][:SID/SERVICE]
     ```

   - Database login information, including schema, user name and password
   - Refresh interval

     Measured in seconds. The default value is 10 seconds.

     You may need to change the refresh interval depending on the amount of logging your system performs. The log consolidator application commits transactions to the database every 10 seconds or when the number of transactions exceeds 80, whichever occurs first.

- Usage and error log settings:
  - Log file names
  - Number of log files for each type of file

  The log file names and number of files must match the actual BIRT iHub configuration.

If you change a BIRT iHub setting, you must also change the corresponding consolidator configuration setting and restart BIRT iHub and the log consolidator application.

The following code example shows the default settings for consolidatorconfig.xml:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LogConsolidator>
   <Database>
      <DriverName>oracle.jdbc.OracleDriver</DriverName>
         <URL>jdbc:oracle:thin:@dbsrv4-w2k:1521:Oran9i</URL>
         <Encoding>UTF8</Encoding>
         <Schema>Users</Schema>
         <DatabasePropertyList>
            <Properties>
               <Name>UserName</Name>
               <Value>actest</Value>
            </Properties>
            <Properties>
               <Name>Password</Name>
               <Value>systest</Value>
            </Properties>
         </DatabasePropertyList>
   </Database>
   <Consolidator>
      <RefreshInterval>10</RefreshInterval>
      <UsageLogEnabled>true</UsageLogEnabled>
      <ErrorLogEnabled>false</ErrorLogEnabled>
      <UsageLogProperties>
         <LogFileName>usage_log</LogFileName>
         <NumberOfLogFiles>2</NumberOfLogFiles>
      </UsageLogProperties>
      <ErrorLogProperties>
         <LogFileName>error_log</LogFileName>
         <NumberOfLogFiles>2</NumberOfLogFiles>
      </ErrorLogProperties>
   </Consolidator>
</LogConsolidator>
```

2. Copy the configuration file, consolidatorconfig.xml, to the following BIRT iHub directory:

   ```
   $AC_SERVER_HOME/etc
   ```

3. Install and run the startup and shutdown files after setting up the database:

   - On a UNIX system, the following Actuate scripts start and stop the consolidator application:

     - start_consolidator.sh

       Starts the log consolidator application. The script takes $AC_SERVER_HOME as an argument and uses it to set the following path variables:

       - Configuration file
       - JAR file directory
       - CLASSPATH

       The script stores the process ID or PID in $AC_SERVER_HOME/etc /consolidator.pid. Add this script to BIRT iHub script start_srvr.sh to start the consolidator application whenever you start BIRT iHub. start_consolidator.sh attempts to start the application five times.

     - stop_consolidator.sh

       Stops the log consolidator application. The script takes $AC_SERVER_HOME as an argument and uses it to read the log consolidator application PID from $AC_SERVER_HOME/etc /consolidator.pid and kills the process. Add this script to BIRT iHub script stop_srvr.sh to stop the consolidator application whenever you stop the BIRT iHub.

   - On a Windows system, the consolidatorwin.exe utility installs and removes the application as a Windows service, and starts and stops the consolidator application. The BIRT iHub installation process installs and runs the utility in the following directory:

     ```
     $AC_SERVER_HOME\bin
     ```

     Use the consolidator.exe utility to install and configure the consolidator application. This utility assumes it is running in the BIRT iHub \bin directory.

     The consolidatorwin.exe utility supports the following command line syntax:

     ```
     consolidatorwin [-H/-?] [-SserviceType] [-UuserName]
        [-Ppassword]
     ```

     The command-line arguments specify the following options:

     - -H requests help on usage

- -S specifies the following types of service:
  - auto adds the log consolidator as the Actuate Usage and Error Logging Consolidator 9 service that starts automatically when Windows restarts.
  - manual adds the log consolidator as the Actuate Usage and Error Logging Consolidator 9 service that requires manual startup when Windows restarts.
  - console starts the consolidator at a Windows command prompt.
  - remove stops the service.
- -U specifies the username
  The Windows user starting the consolidator application. Actuate recommends using the same user as the user that starts BIRT iHub.
- -P specifies the password
  The user's password.

The following command adds the consolidator application as a Windows service that starts automatically when Windows starts:

```
consolidatorwin -Sauto -UUsername -PPassword
```

How to configure the log consolidator database

1. Configure a database server machine, Oracle server, and database.
2. Using Oracle SQL*Plus, log in as the system database administrator.
3. Run the CreateActuateLogTables.sql script.

   The following command runs the CreateActuateLogTables.sql script from the default directory in the BIRT iHub Integration Technology installation for Windows:

```
SQL> @"C:\Program Files\Actuate\ServerIntTech
   \UsageAndErrorConsolidator\DBScripts
   \CreateActuateLogTables.sql";
```

CreateActuateLogTables.sql drops the ActuateLog and ActuateLogUser users, performs cascading deletes on all their objects, including sequences, tables, and indexes, and recreates these objects. The script creates the following database objects:

- Tables to contain the usage and error log data
  INSERT statements add predefined codes and descriptions for the event, file, job, object, operation, output format, service, and status types, after creating the tables.

- Sequence generators to provide the values for usage and error event IDs when inserting log data

  Each usage and error event ID is a unique value.

- Indexes to contain primary and foreign key columns in the tables:
  - Primary key constraints on columns containing predefined codes ensure that these values are unique and not null.
  - Foreign key references on columns containing predefined codes ensure that these values are consistent with the values in the primary keys.

The ActuateLog schema contains the following tables and indexes:

- AcAdminEvent

  Contains the log records for administration operation events, including the following data:
  - Event ID
  - Object type code, indicating a User, Role, Channel, Group, File, or Folder object
  - Object operation code, indicating a Create, Delete, Modify, Login, Logout, or Download operation
  - Object name, version name, size, and attribute
  - Old and new values

  Table 11-7 shows the structure of the AcAdminEvent table.

Table 11-7 Structure of the AcAdminEvent table

Column	Data type	Constraint	References	Key
EventId	INTEGER	AcApEv_AdEvId_Idx	AcEvent.EventId	Primary
ObjectTypeCode	INTEGER		AcObjectType.ObjectTypeCode	
ObjectOperationCode	INTEGER		AcObjectOperation.ObjectOperationCode	
ObjectName	VARCHAR2(1000)			
ObjectVersionName	VARCHAR2(255)			

Table 11-7 Structure of the AcAdminEvent table

Column	Data type	Constraint	References	Key
ObjectSize	INTEGER			
ObjectAttribute	VARCHAR2(50)			
OldValue	VARCHAR2(2000)			
NewValue	VARCHAR2(2000)			

- AcApplicationEvent
  Contains the log records for application events, including the following data:
  - Event ID
  - Executable name
  - Executable version, indicating an RPTDOCUMENT, RPTDESIGN, or other file type
  - Job type code, indicating an Async, Persistent, and Transient job type
  - Resource group ID
  - Dispatch node, indicating the volume, system, and server
  - Output format code, indicating PDF, XLS, HTML, or other output format

  Table 11-8 shows the structure of the AcApplicationEvent table.

Table 11-8 Structure of the AcApplicationEvent table

Column	Data type	Constraint	References	Key
EventId	INTEGER	AcApEv _ApEvId _Idx	AcEvent .EventId	Primary
ExecutableName	VARCHAR2(1000)			
ExecutableVersion	VARCHAR2(100)		AcFileType .FileTypeCode	
FileTypeCode	INTEGER			
Parameters	VARCHAR2(2000)			
JobName	VARCHAR2(100)			
JobTypeCode	INTEGER		AcJobType .JobTypeCode	
JobSubmittedTimestamp	DATE			

(continues)

Table 11-8 Structure of the AcApplicationEvent table (continued)

Column	Data type	Constraint	References	Key
ResourceGroupId	INTEGER		AcResource Group .Resource GroupId	
DispatchNode	INTEGER		AcSystem Component .System ComponentId	
RequestId	VARCHAR2(100)			
RequestWaitTime	INTEGER			
RequestRunningTime	INTEGER			
OutputName	VARCHAR2(1000)			
OutputVersion	VARCHAR2(100)			
OutputFormatCode	INTEGER		AcOutput Format .Output Format Code	
OutputSize	INTEGER			
PageCount	INTEGER			
PageNumbersViewed	VARCHAR2(50)			

- AcErrorEvent
  Contains the log records for error events, including the following data:
  - Error event ID
  - System component ID, indicating the volume, system, and server
  - User name
  - Error code, category, severity, parameters, and message

  Table 11-9 shows the structure of the AcErrorEvent table.

Table 11-9 Structure of the AcErrorEvent table

Column	Data type	Constraint	References	Key
ErrorEventId	INTEGER	AcErEv _ErEvId _Idx	AcEvent .EventId	Primary

Table 11-9 Structure of the AcErrorEvent table

Column	Data type	Constraint	References	Key
EventTimestamp	DATE			
SystemComponentId	INTEGER		AcSystem Component .System ComponentId	
UserName	VARCHAR2(255)			
ErrorCode	INTEGER			
ErrorCategory	VARCHAR2(255)			
ErrorSeverity	INTEGER			
ErrorParameter1	VARCHAR2(50)			
ErrorParameter2	VARCHAR2(50)			
ErrorParameter3	VARCHAR2(50)			
ErrorMessage	VARCHAR2(255)			

- AcErrorLogOffset
  Contains the following usage log data:
  - File offsets
  - Volume names
  - Last update timestamp

  Table 11-10 shows the structure of the AcErrorLogOffset table.

Table 11-10 Structure of the AcErrorLogOffset type

Column	Data type/Values	Constraint	References	Key
FileIndex	NUMBER			
FileOffset	NUMBER			
VolumeName	VARCHAR2(50)	NOT NULL		Primary
LastUpdateTimeStamp	NUMBER			

- AcEvent
  Contains the log records for events, including the following data:
  - Event ID
  - Event timestamp
  - System component ID, indicating the volume, system, and server

- Event type code, indicating a Generate, Print, View, Delete, Admin, Query, or Search event type
- Start and end timestamps
- Status code, indicating Success or Failure
- Service type code, indicating a Factory, View, Encyclopedia, Integration, or Cache service type

Table 11-11 shows the structure of the AcEvent table.

Table 11-11 Structure of the AcEvent table

Column	Data type	Constraint	References	Key
EventId	INTEGER	AcSt_StCo_Idx		Primary
EventTimestamp	DATE			
SystemComponentId	INTEGER		AcSystem Component .SystemComponent Id	
UserName	VARCHAR2(50)			
EventTypeCode	INTEGER		AcEventType .EventTypeCode	
StartTimestamp	DATE			
EndTimestamp	DATE			
StatusCode	INTEGER		AcStatus .StatusCode	
ServiceTypeCode	INTEGER	EventId	AcServiceType .ServiceType Code	

- AcEventType
  Contains the codes and descriptions for event types, including the following data:
  - Event type code, including the following predefined values:

    `1 through 7`
  - Event type description, including the following predefined values:

    `Generate, Print, View, Delete, Admin., Query, Search`

Table 11-12 shows the structure of the AcEventType table.

Table 11-12 Structure of the AcEventType table

Column	Data type/Values	Constraint	References	Key
EventTypeCode	INTEGER1-7	AcEvTy_EvTyCo_Idx		Primary
EventTypeDescription	VARCHAR2(50)			

- AcFileType
  Contains the codes and descriptions for the following file type data:
  - File type codes, including the following predefined values:

    `1 through 39`
  - File types, including the following predefined values:

    ```
    UNKNOWN, DOX, DCD, HTM, HTML, ICD, IOB, PDF, ROL, ROP, ROS,
       ROV, ROW, RPTDESIGN, RPTDOCUMENT, RPTLIBRARY, RPTTEMPLATE,
       RTF, SMA, TXT, XLS
    ```

  Table 11-13 shows the structure of the AcFileType table.

Table 11-13 Structure of the AcFileType table

Column	Data type/Value	Constraint	References	Key
FileTypeCode	INTEGER1-39	AcFiTy_FiTyCo_Idx		Primary
FileType	VARCHAR2(20)			

- AcJobType
  Contains the codes and descriptions for the following job type data:
  - Job type codes, including the following predefined values:

    `1 through 3`
  - Job type descriptions, including the following predefined values:

    `Async, Persistent, Transient`

  Table 11-14 shows the structure of the AcJobType table.

Table 11-14 Structure of the AcJobType table

Column	Data type/Value	Constraint	References	Key
JobTypeCode	INTEGER1-3	AcJoTy_JoTyCo_Idx		Primary
JobTypeDescription	VARCHAR2(20)			

- AcObjectOperation
  Contains the codes and descriptions for the following object operation data:
  - Object operation codes, including the following predefined values:

    `1 through 6`
  - Object operation descriptions, including the following predefined values:

    `Create, Delete, Modify, Login, Logout, Download`

  Table 11-15 shows the structure of the AcObjectOperation table.

Table 11-15 Structure of the AcObjectOperation table

Column	Data type/Value	Constraint	References	Key
ObjectOperationCode	INTEGER	AcObOp_ObOpCo_Idx		Primary
ObjectOperation Description	VARCHAR2(20)			

- AcObjectType
  Contains the codes and descriptions for the following object type data:
  - Object type codes, including the following predefined values:

    `1 through 6`
  - Object type descriptions, including the following predefined values:

    `User, Role, Channel, Group, File, Folder`

  Table 11-16 shows the structure of the AcObjectType table.

Table 11-16 Structure of the AcObjectType table

Column	Data type/Values	Constraint	References	Key
ObjectTypeCode	INTEGER	AcObTy_ObTyCo_Idx		Primary
ObjectTypeDescription	VARCHAR2(20)			

- AcOutputFormat
  Contains the codes and descriptions for the following output format data:
  - Output format codes, including the following predefined values:

    `1 through 42`
  - Output format descriptions, including the following predefined values:

    ```
    UNKNOWN, PDF, XLS, ROW, DHTML, HTML, HTM, RTF, REPORTLET,
       XMLDISPLAY, XMLCOMPRESSEDDISPLAY, DHTMLRAW, DHTMLLONG,
       CSS, ANALYSIS, EXCELDISPLAY, EXCELDATA, EXCELDATADUMP,
       RTFFULLYEDITABLE, UNCSV, TSV, EXCEL, XMLDATADUMP,
       XMLREPORTLET, XMLCOMPRESSEDREPORTLET, XMLCOMPRESSEDEXCEL,
    ```

```
XMLCOMPRESSEDPDF, XMLCOMPRESSEDRTF, XMLSTYLE, XMLDATA,
RPTDOCUMENT, RPTLIBRARY, RPTTEMPLATE
```

Table 11-17 shows the structure of the AcOutputFormat table.

Table 11-17 Structure of the AcOutputFormat table

Column	Data type/Values	Constraint	References	Key
OutputFormatCode	INTEGER1-42	AcOuFo_OuFoCo_Idx		Primary
OutputFormat Description	VARCHAR2(30)			

- AcResourceGroup
  Contains the codes and descriptions for the following resource group data:
  - Resource group ID, including the following predefined value:

    `0`
  - Resource group name, including the following predefined value:

    `NULL`

  Table 11-18 shows the structure of the AcResourceGroup table.

Table 11-18 Structure of the AcResourceGroup table

Column	Data type/Values	Constraint	References	Key
ResourceGroupId	INTEGER	AcReGr_ReGrId_Idx		Primary
ResourceGroupName	VARCHAR2(128)			

- AcServiceType
  Contains the codes and descriptions for the following service type data:
  - Service type code, including the following predefined values:

    `1 through 5`
  - Service type description, including the following predefined values:

    `Factory, View, Encyclopedia, Integration, Cache`

  Table 11-19 shows the structure of the AcServiceType table.

Table 11-19 Structure of the AcServiceType table

Column	Data type/Values	Constraint	References	Key
ServiceTypeCode	INTEGER	AcSeTy_SeTyCo_Idx		Primary
ServiceTypeDescription	VARCHAR2(50)			

- AcStatus
  Contains the codes and descriptions for the following status type data:
  - Status codes, including the following predefined values:

    `0 through 1`
  - Status descriptions, including the following predefined values:

    `Failure, Success`

  Table 11-20 shows the structure of the AcStatus table.

Table 11-20 Structure of the AcStatus type

Column	Data type/ Values	Constraint	References	Key
StatusCode	INTEGER	AcSt_StCo _Idx		Primary
StatusDescription	VARCHAR2(20)			

- AcSystemComponent
  Contains the log records for system components, including the following data:
  - System component ID
  - Volume, system, and server names

  Table 11-21 shows the structure of the AcSystemComponent table.

Table 11-21 Structure of the AcSystemComponent table

Column	Data type/ Values	Constraint	References	Key
SystemComponentId	INTEGER	AcSt_StCo _Idx		Primary
VolumeName	VARCHAR2(50)			
SystemName	VARCHAR2(50)			
ServerName	VARCHAR2(50)			

- AcUsageLogOffset
  Contains the following usage log data:
  - File offsets
  - Volume names
  - Last update timestamp

Table 11-22 shows the structure of the AcUsageLogOffset table.

Table 11-22 Structure of the AcUsageLogOffset type

Column	Data type/ Values	Constraint	References	Key
FileIndex	NUMBER,			
FileOffset	NUMBER			
VolumeName	VARCHAR2(50)	NOT NULL		Primary
LastUpdateTime Stamp	NUMBER			

The ActuateLog schema contains the following indexes listed in Table 11-23.

Table 11-23 Indexes in the ActuateLog schema

Index	Table.Column(s)
AcApEv_AdEvId_Idx	AcAdminEvent.EventId
AcApEv_ApEvId_Idx	AcApplicationEvent.EventId
AcApEv_ExNa_Idx	AcApplicationEvent.ExecutableName
AcApEv_FiTyCo_Idx	AcApplicationEvent.FileTypeCode
AcApEv_JoTyCo_Idx	AcApplicationEvent.JobTypeCode
AcAdEv_ObNa_Idx	AcAdminEvent.ObjectName
AcApEv_OuNa_Idx	AcApplicationEvent.OutputName
AcAdEv_ObTyCo_ObOpCo_Idx	AcAdminEvent.ObjectTypeCode, ObjectOperationCode
AcErEv_ErCo_Idx	AcErrorEvent.ErrorCode
AcErEv_ErEvId_Idx	AcErrorEvent.ErrorEventId
AcErEv_ErSe_Idx	AcErrorEvent.ErrorSeverity
AcErEv_EvTi_Id	AcErrorEvent.EventTimestamp
AcErEv_SyCoId_Idx	AcErrorEvent.SystemComponentId
AcErEv_UsNa_Idx	AcErrorEvent.UserName
AcEv_EvId_Idx	AcStatus.EventId
AcEv_SyCoId_Idx	AcEvent.SystemComponentId
AcEv_StCo_Idx	AcEvent.StatusCode
AcEv_SyTyCo_Idx	AcEvent.ServiceTypeCode
AcEv_EvTi_Idx	AcEvent.EventTimestamp

(continues)

Table 11-23 Indexes in the ActuateLog schema (continued)

Index	Table.Column(s)
AcEv_EvTyCo_Idx	AcEvent.EventTypeCode
AcEv_UsNa_Idx	AcEvent.UserName
AcEvTy_EvTyCo_Idx	AcEventType.EventTypeCode
AcFiTy_FiTyCo_Idx	AcFileType.FileTypeCode
AcJoTy_JoTyCo_Idx	AcJobType.JobTypeCode
AcObOp_ObOpCo_Idx	AcObjectOperation.ObjectOperationCode
AcObTy_ObTyCo_Idx	AcObjectType.ObjectTypeCode
AcOuFo_OuFoCo_Idx	AcOutputFormat.OutputFormatCode
AcReGr_ReGrId_Idx	AcResourceGroup.ResourceGroupId
AcSeTy_SeTyCo_Idx	AcServiceType.ServiceTypeCode
AcSt_StCo_Idx	AcStatus.StatusCode
AcSyCo_SyCoId_Idx	AcSystemComponent.SystemComponentId

Chapter

12

Actuate logging and monitoring functions

This chapter provides an alphabetical list of the functions of the Actuate Usage Logging, Error Logging, and Performance Monitoring Extensions. It contains the following topics:

- About Usage Logging Extension functions
- About Error Logging Extension functions
- About the Performance Monitoring API

About Usage Logging Extension functions

The Usage Logging Extension supports retrieving usage information that BIRT iHub captures and writing that information to a log file.

AcIsThreadSafe

Specifies whether the Usage Logging Extension is multithread-safe. If the extension is not multithread-safe, AcIsThreadSafe must return False and BIRT iHub serializes all calls to the extension.

Syntax

```
USAGELOGEXT_API int AcIsThreadSafe ( );
```

AcLogUsage

Called by BIRT iHub for every transaction it logs.

Syntax

```
USAGELOGEXT_API void AcLogUsage (UsageInformation* usagenfo);
```

Parameters

usageInfo
The usage information that BIRT iHub captures.

AcStartUsageLog

Specifies the path to the usage log file and the BIRT iHub to monitor. For example, if you write the usage information to a database, AcStartUsageLog provides a placeholder to open the database connection.

The server and cluster parameters are WideChar pointers. The function uses the data type WideChar for Unicode compatibility. Actuate pushes out all string data in UCS-2 format.

Syntax

```
USAGELOGEXT_API int  AcStartUsageLog (const char* serverHome, const
    WideChar* server, const WideChar* cluster);
```

Parameters

serverHome
The path to the usage log file. The path must be $AC_SERVER_HOME/bin.

server
The name of the BIRT iHub to monitor.

cluster
The name of the cluster of which the BIRT iHub is a member.

AcStopUsageLog

Stops logging usage information and releases resources.

Syntax USAGELOGEXT_API void AcStopUsageLog ( );

About Error Logging Extension functions

This section provides an alphabetical list of the Error Logging Extension functions. Each entry includes a general description of the function, its syntax, and a description of its parameters.

The Error Logging Extension supports retrieving error information that BIRT iHub captures and writing that information to a log file.

AcIsThreadSafe

Indicates whether the Error Logging Extension is multithread-safe. If the extension is not multithread-safe, AcIsThreadSafe must return False and BIRT iHub serializes all calls to the extension.

Syntax ERRORLOGEXT_API int AcIsThreadSafe ( );

AcLogError

Called by BIRT iHub for every error it encounters.

Syntax ERRORLOGEXT_API void AcLogError (ErrorInformation* errorInfo);

Parameters

errorInfo
The error information that BIRT iHub captures.

AcStartErrorLog

Specifies the path to the error log file and the BIRT iHub to monitor. For example, if you write the error information to a database, AcStartErrorLog provides a placeholder to open the database connection.

The server and cluster parameters are WideChar pointers. The function uses the data type WideChar for Unicode compatibility. Actuate pushes out all string data in UCS-2 format.

Syntax ERRORLOGEXT_API int AcStartErrorLog (const char* serverHome, const WideChar* server, const WideChar* cluster);

Parameters

serverHome
The path to the error log file. The path must be $AC_SERVER_HOME/bin.

server
The name of the BIRT iHub to monitor.

cluster
The name of the cluster of which the BIRT iHub is a member.

AcStopErrorLog

Stops logging error information and releases resources.

Syntax ERRORLOGEXT_API void AcStopErrorLog ( );

About the Performance Monitoring API

This section provides an alphabetical list of the Performance Monitoring Extension API operations and data types. Each entry includes a general description of the operation, its schema, and a description of its elements.

The Performance Monitoring Extension supports monitoring various server counters by Windows perfmon. These functions are found within the WSDL.

ArrayOfCounterInfo

A complex data type that represents an array of CounterInformation objects.

Schema

```
<xsd:complexType name="ArrayOfCounterInfo">
   <xsd:sequence>
      <xsd:element name="CounterInfo" type="typens:CounterInfo"
         minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

CounterInfo

A complex data type that describes a counter.

Schema

```
<xsd:complexType name="CounterInfo">
   <xsd:sequence>
      <xsd:element name="CounterId" type="xsd:long"/>
      <xsd:element name="CounterName" type="xsd:string"/>
      <xsd:element name="CounterValue" type="xsd:long"/>
   </xsd:sequence>
</xsd:complexType>
```

Elements

CounterId
The ID of the counter.

CounterName
The name of the counter.

CounterValue
The value of the counter.

GetAllCounterValues

Retrieves the values of all counters.

Request schema

```
<xsd:complexType name="GetAllCounterValues"/>
```

Response schema

```
<xsd:complexType name="GetAllCounterValuesResponse">
<xsd:complexType>
   <xsd:sequence>
      <xsd:element name="CounterInfoList"
         type="typens:ArrayOfCounterInfo"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

CounterInfoList
The values of all counters.

GetCounterValues

Retrieves the IDs, names, and values of specific counters.

Request schema

```
<xsd:complexType name="GetCounterValues">
   <xsd:sequence>
      <xsd:element name="CounterIDList" type="typens:ArrayOfLong"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

CounterIDList
The list of counters for which to retrieve information.

Response schema

```
<xsd:complexType name="GetCounterValuesResponse">
   <xsd:sequence>
      <xsd:element name="CounterInfoList"
         type="typens:ArrayOfCounterInfo"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

CounterInfoList
The IDs, names, and values of the specified counters.

ResetCounters

Resets the values of the specified logging and monitoring counters to zero.

Request schema

```
<xsd:complexType name="ResetCounters">
   <xsd:sequence>
      <xsd:element name="CounterIDList" type="typens:ArrayOfLong"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

CounterIDList
The list of counters to reset.

Response schema

```
<xsd:element name="ResetCountersResponse"/>
```

Chapter

13

Aging and archiving Encyclopedia volume items

This chapter contains the following topics:

- Automating report archival and removal
- Aging and archiving an item using the Actuate Information Delivery API

Automating report archival and removal

Reports, folders, and other items accumulate in an Encyclopedia volume unless you archive or remove them. Archiving is the process of placing a file in an archive directory. Aging an item means setting a limit on the length of time to retain the file before deleting it from the volume. When you set aging rules, you indicate whether to archive the item before deleting it. Expiring a file means removing it from the volume. When a file expires, the system either deletes it or places it in the archive directory, depending on the file's archive rules.

The Actuate Information Delivery API supports automating the aging and archiving processes for Encyclopedia volume items. Autoarchiving is the process of archiving and removing items on a specific schedule, using the archive rules that you create.

You set autoarchive rules when you create or update report files, folders, and job completion notices. For example, you can create a rule that removes all quarterly sales reports older than one year, or a rule that removes all successful job notifications older than ten days, or one that archives the oldest instance of a daily stock report when the volume contains more than five newer instances.

The Encyclopedia volume administrator creates the archive to hold the archived files. Each volume has a single archive.

About Actuate Online Archive Driver

Archiving files and folders requires installation of an archive driver. You configure a single archive driver for an Encyclopedia volume. When BIRT iHub archives a file in an Encyclopedia volume, BIRT iHub creates a copy of the file, then sends the copy to the Actuate archive driver. The driver creates an archive that retains the same folder structure as the Encyclopedia volume.

Actuate Online Archive Driver is the Java application that copies an archive to another Encyclopedia volume. This application copies expired Encyclopedia volume files to the second Encyclopedia volume that serves as a file archive. The archive created by the Online Archive Driver remains accessible as a folder in BIRT iHub System.

The Actuate Online Archive Driver IDAPI provides a SOAP-based interface between BIRT iHub and external archive software. Actuate iServer Release 8 and later, as well as BIRT iHub, provide a reference implementation of Actuate Online Archive Driver in the Online Archive Driver folder of BIRT iHub Integration Technology.

Using this application requires the Online Archive option for BIRT iHub System. You must have this option installed on BIRT iHub to use the driver or the source code.

Configuring the Online Archive Driver

The Online Archive Driver implementation supports retaining file attributes in an archive. In the Java application's XML configuration file, onlinearchive.cfg, you can specify that an archive retains its file attributes using the following settings:

- RetainTimestamp
  Specifies whether to retain the time stamp. The default is False.
- RetainOwner
  Specifies whether to retain the owner. The default is False.

  The online archive application attempts to match any user or role in the file's access control list (ACL) with the same name in the archive Encyclopedia volume.
- RetainPermission
  Specifies whether to preserve the permissions in the access control list (ACL). The default is False.

  If you do not archive the access control list (ACL) of a file, the file owner has full access. If the user or role does not exist in the archive Encyclopedia volume, you can configure the application to create a user or role with the same name.
- CopyDependOnFile
  Specifies whether to copy the dependent files for the archive file. The default is True.

  If the online archive application archives a file that has a dependency on another file, the application archives both files. The application does not delete the file on which the archived file has a dependency unless the application is archiving both files.

In the configuration file, you can also specify the following options:

- CreateUserRole
  Specifies whether to create missing user or roles in the target volume in order to retain the owner or permissions. The default is True.

  The online archive application does not enable a login for a user or role it creates.
- ArchiveRoot
  Specifies the root encyc folder for all archived files. The default is /.
- CreateArchiveSubFolder
  Specifies whether to create the archive as a subfolder under the root folder that contains a time stamp. The default is True.

The online archive application creates a folder in the archive Encyclopedia volume and places the files from the archive session into the subfolder. Within the archive, the archive driver creates a folder hierarchy identical to the one in the Encyclopedia volume and places the files in the hierarchy.

The archive root folder name contains the start date and time of the archive session in the following format:

```
YYYY_mm_dd.hh_mm_ss
```

For example, if you optionally specify the name of the root archive folder in the configuration files as /MyArchive and the archive sessions starts at 6:00 A.M. May 31, 2008, the application copies the archived files to the following folder:

```
/MyArchive2008_05_31.06_00_00
```

If you set CreateArchiveSubFolder to False, the online archive application suppresses the creation of the time stamp folder and does not create the archive as a subfolder. The archive volume mimics the folder structure of the production volume. The CreateArchiveSubFolder option preserves the version numbers of the archived files by placing the archive folder in a separate root folder that has a time stamp.

- LogLevel

  Specifies the level of detail in log file. Valid values are Summary, Detail, and Trace. The default is Summary.

BIRT iHub installs an onlinearchive.cfg file in the following location:

```
$AC_SERVER_HOME\etc
```

Actuate Server Integration Technology also provides a copy of onlinearchive.cfg in the Online Archive Driver folder. The following example shows the onlinearchive.cfg code:

```
<?xml version="1.0" encoding="UTF-8"?>
<archiveconfig>
   <!-- ACTUATE ONLINE ARCHIVE DRIVER CONFIGURATION FILE     -->
   <!-- See README.html in the onlinearchive directory in    -->
   <!-- Server Integration Technology installation for usage -->
   <!-- and licensing information                            -->

   <!--TargetServer, TargetSOAPPort: [Required]              -->
   <!--    Name or IP of server and port for connecting to   -->
   <!--    the SOAP dispatcher service of the target volume  -->
   <TargetServer>127.0.0.1</TargetServer>
   <TargetSOAPPort>8000</TargetSOAPPort>
   <!--ArchiveVolume: [Required]                             -->
   <!--    Name of target volume to copy archived files to   -->
   <ArchiveVolume>DefaultVolume</ArchiveVolume>
```

```
  <!--AdminUser, AdminPassword: [Required]                  -->
  <!--    Name and password of a user in the target volume  -->
  <!--    that belongs to the Administrator role            -->
  <AdminUser>administrator</AdminUser>
  <AdminPassword></AdminPassword>

  <!--RetainTimestamp: [Optional, default: false]           -->
  <!--    Whether timestamp of archived file is preserved   -->
  <RetainTimestamp>false</RetainTimestamp>

  <!--RetainOwner: [Optional, default: false]               -->
  <!--    Whether Owner of archived file is preserved       -->
  <RetainOwner>false</RetainOwner>

  <!--RetainPermission: [Optional, default: false]          -->
  <!--    Whether Permission (ACL) of archived file is      -->
  <!--    preserved                                         -->
  <RetainPermission>false</RetainPermission>

  <!--CopyDependOnFile: [Optional, default: true]           -->
  <!--    Whether files depended on by archived file are    -->
  <!--    copied                                            -->
  <CopyDependOnFile>true</CopyDependOnFile>

  <!--CreateUserRole: [Optional, default: true]             -->
  <!--    Whether to create missing user or roles in target -->
  <!--    volume in order to retain Owner or Permissions    -->
  <CreateUserRole>true</CreateUserRole>

  <!--ArchiveRoot: [Optional, default: /]                   -->
  <!--    Root encyc folder for all archived files          -->
  <ArchiveRoot>/</ArchiveRoot>

  <!--CreateArchiveSubFolder: [Optional, default: true]     -->
  <!--    Whether to create a timestamp dependent subfolder -->
  <!--    under ArchiveRoot for each archive session        -->
  <CreateArchiveSubFolder>true</CreateArchiveSubFolder>

  <!-- LogLevel: [Optional, default: Summary]               -->
  <!--    Leve of detail in log file. Valid values are:     -->
  <!--    Summary, Detail and Trace -->
  <LogLevel>Summary</LogLevel>

</archiveconfig>
```

How to install an online archive configuration file

1. Copy and rename the $AC_SERVER_HOME\etc \onlinearchive.cfg file to contain the name of the Encyclopedia volume, using the following format:

   ```
   onlinearchive_<volume>.cfg
   ```

The configuration file name looks like the following example:

```
onlinearchive_enl2509.cfg
```

2 Edit the online archive configuration file by performing the following tasks:

   1 Navigate to the following code:

```
<!-- ArchiveVolume: [Required] -->
<!-- Name of target volume to copy archived files to -->
<ArchiveVolume>DefaultVolume</ArchiveVolume>
```

   2 Change the parameter for <ArchiveVolume> from DefaultVolume to the name of the Encyclopedia volume.

   The code now looks like the following example:

```
<!-- ArchiveVolume: [Required] -->
<!-- Name of target volume to copy archived files to -->
<ArchiveVolume>enl2509</ArchiveVolume>
```

   3 Save and close onlinearchive_<volume>.cfg.

To complete the installation, you must configure the online archive service provider in the Encyclopedia volume. BIRT iHub installs a shell script for starting the online archive service in the following location:

```
$AC_SERVER_HOME\bin
```

In a Windows installation, the name of the script is aconlinearchive.bat. In a UNIX or Linux installation, the name of the script is aconlinearchive.sh. These scripts configure the Java application run-time environment for the archive driver by performing the following tasks:

- Specifies the Java Runtime Environment (JRE) by setting the environment variable, ARCHIVE_DRIVER_JRE, to the BIRT iHub default JRE specified by $AC_JRE_HOME.

  You can use a different JRE, but this version is the only JRE version which has been tested.

- Specifies the class path for Online Archive Driver JAR file by setting the environment variable, DRIVER_JAR_PATH to %AC_SERVER_HOME% \drivers.

  BIRT iHub installs aconlinearchive.jar, the application library, and aconlinearchiveDEP.jar, the dependent library files, in $AC_SERVER_HOME \drivers.

- Starts the online archive service by making a call to execute the class, com.actuate.onlinearchivedriver.Main.

How to configure the volume archive service provider

1 Log in to the iHub Configuration Console and choose Advanced view.

2 From the side menu, choose Volumes.

3 On Volumes, select the icon to the left of the Encyclopedia volume. Choose Properties.

4 On Volume—Properties, perform the following tasks:

5 In Volume archive service provider, select Use command line. Type:

```
aconlinearchive.bat
```

Volume—Properties looks like Figure 13-1.

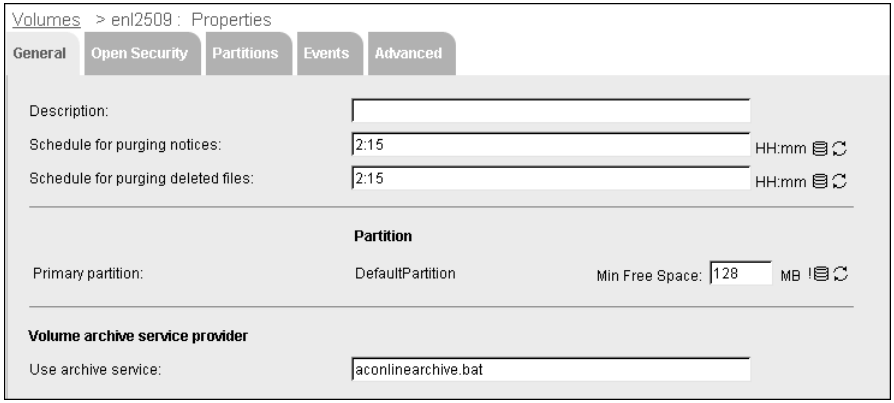


Figure 13-1 Configuring the volume archive service provider

Choose OK.

6 Log out of iHub Configuration Console.

Understanding aging and archiving rules for items in an Encyclopedia volume

You can set an expiration policy for a folder that affects all items in that folder. Alternatively, you can set the policy for an individual report.

An item ages and expires according to a set of rules you apply to the item itself, to the folder that contains it, or to the entire Encyclopedia volume. The following aging and archiving rules apply to Encyclopedia volume items:

- Volume archiving rules apply to every folder in the volume, including subfolders. By default, the Encyclopedia volume archives files and folders once a day. The system administrator can change this default setting to specify when and how often to archive files and folders.
- Folder archiving rules apply to the entire hierarchy of reports in the folder, unless subfolders also have age or date properties.

- Subfolder archiving rules supersede age or date properties of folders higher in the hierarchy.
- A rule for a file overrides a rule inherited from the folder that contains the file.
- Folder aging and archiving properties specify the file type to which those properties apply. You can specify file types explicitly or use default values.
- The aging process does not remove folders during archival, only their contents.
- Archive rules determine whether the system ages dependent files along with the original files.

Understanding precedence in archiving

The autoarchive rule for a file takes precedence, if the file has a rule. If a file does not have an autoarchive rule, Actuate applies the next available autoarchive rule in the following order of precedence:

- The file's autoarchive rules take precedence.
- If the file has no autoarchive rules, the system applies the rules for the file type. You set file type rules for at the folder level, so the system looks for the file type rules in the file's folder hierarchy.
- If there are no rules for a file type in the file's folder hierarchy, the system applies the general autoarchive rule for folders in the file's folder hierarchy.
- If there are no general autoarchive rules for folders in the file's folder hierarchy, the system applies the Encyclopedia volume's settings.

Aging and archiving an item using the Actuate Information Delivery API

Archiving and aging rules apply to BIRT reports, third-party reports, folders, and job notifications. Using the Actuate Information Delivery API, an application developer can create, update, view, and remove autoarchive rules for these items.

A developer also can set autoarchive rules for all the files in an Encyclopedia volume or for a particular file or file type.

A user running the application must have delete and write privileges to modify autoarchive settings.

Setting and updating autoarchive rules using IDAPI

Use the following Actuate Information Delivery API operations to set autoarchive rules programmatically:

- SubmitJob and UpdateFile

  To set archive rules when you use SubmitJob, set one or more elements of ArchiveRule.

  To add or update autoarchive rules for an existing file or folder, use the SetArchiveRules, AddArchiveRules, or RemoveArchiveRules suboperations of UpdateFile. Each of these suboperations has an ArchiveRule element.

  You can set the following elements of ArchiveRule, described in Table 13-1.

Table 13-1 ArchiveRule elements

Element	Description
ArchiveOnExpiration	True if you want to archive the file before the system deletes it.
ExpirationAge	The number of minutes before expiration. If you set ExpirationAge, you cannot set ExpirationTime.
ExpirationTime	The time of day for the expiration. If you set ExpirationTime, you cannot set ExpirationAge.
ExpireDependentFiles	True if dependent files expire with the original file.
FileType	The file type to which the rule applies. To set the rule for a folder and its subfolders, specify Directory.
InheritedFrom	The source of an inherited rule.
IsInherited	True if the item inherits archiving rules. If this element is True, the system ignores all other elements except FileType.
NeverExpire	True if the file does not have an expiration date or expiration age. If this element is True, the system ignores ExpirationAge and ExpirationTime.

- UpdateJobSchedule

  Use the SetParameters suboperation to change the autoarchive rules for the job output file. You can change the following archive-related elements of SetParameters, described in Table 13-2.

Table 13-2 Archive-related elements of SetParameters

Element	Description
ArchiveOnExpire	True if you want to archive the file before the system deletes it
ArchiveRuleInherited	Indicates whether the job output file inherits archive rules
ExpirationAge	The number of minutes before the output file expires
ExpirationDate	The date on which the output file expires
NeverExpire	True if the file does not have an expiration date or expiration age

- UpdateVolumeProperties
  Use the SetAutoArchiveSchedules suboperation to update the autoarchive schedule details for the volume.

Setting default autoarchive rules when creating a folder

Use CreateFolder followed by UpdateFile to set autoarchive rules for a folder when you create the folder. In UpdateFile, set the ArchiveRule element of SetArchiveRules. In the following example, FileType $$$ indicates that the file is a folder. The autoarchive rules you set in the request apply to all file types in the folder except those that already have archive rules. When NeverExpire is False, you must set either an expiration date or an expiration age. Express the expiration age in minutes.

```
<SOAP-ENV:Body>
   <Administrate>
      <Transaction>
         <CreateFolder>
            <FolderName>/Inventory/Timeshares</FolderName>
            <IgnoreDup>false</IgnoreDup>
         </CreateFolder>
         <UpdateFile>
            <SetArchiveRules>
               <ArchiveRule>
                  <FileType>$$$</FileType>
                  <NeverExpire>false</NeverExpire>
                  <ArchiveOnExpiration>true</ArchiveOnExpiration>
                   <ExpirationAge>14400</ExpirationAge>
                  <IsInherited>false</IsInherited>
               </ArchiveRule>
            </SetArchiveRules>
            <Name>/Inventory/Timeshares</Name>
         </UpdateFile>
      </Transaction>
   </Administrate>
</SOAP-ENV:Body>
```

As with other Administrate operations, the response to this request is empty when the operation succeeds. In the event of failure, the response is an error message.

Setting autoarchive rules when creating a job schedule

Set properties of the ArchiveRule element, as shown in the following example, to set autoarchive rules for a report when you create a schedule for the report. If you set an expiration age, express the age in minutes. For example, to set an expiration age of 30 days, express the age as 43200 minutes. If you want the file to inherit

archive rules from a file or folder, set IsInherited to True and use InheritedFrom to show the path to the file or folder from which the file inherits its rules.

```
<SOAP-ENV:Body>
   <SubmitJob>
      <JobName>pie</JobName>
      <Priority>500</Priority>
      <InputFileName>/Inventory/pie.rptdesign</InputFileName>
      <RunLatestVersion>true</RunLatestVersion>
      <RequestedOutputFile>
         <Name>/Inventory/pie.rptdocument</Name>
         <ReplaceExisting>false</ReplaceExisting>
         <ArchiveRule>
            <FileType>rptdocument</FileType>
            <ArchiveOnExpiration>true</ArchiveOnExpiration>
            <ExpirationAge>43200</ExpirationAge>
            <IsInherited>false</IsInherited>
         </ArchiveRule>
         …
   </SubmitJob>
</SOAP-ENV:Body>
```

The response to this operation is the JobId if the job succeeds. If the job fails, an error message appears.

Updating autoarchive rules for a file or folder

Use UpdateFile and modify the ArchiveRule property of the SetArchiveRules element to update the autoarchive rules for a file or folder. Express the expiration age in minutes.

```
<SOAP-ENV:Body>
   <Administrate>
      <UpdateFile>
         <SetArchiveRules>
            <ArchiveRule>
               <FileType>RPTDESIGN</FileType>
               <NeverExpire>false</NeverExpire>
               <ArchiveOnExpiration>true</ArchiveOnExpiration>
               <ExpirationAge>28800</ExpirationAge>
            </ArchiveRule>
         </SetArchiveRules>
         <IdList>
            <String>4</String>
         </IdList>
      </UpdateFile>
   </Administrate>
</SOAP-ENV:Body>
```

As with other Administrate operations, the response to this request is empty when the operation succeeds. In the event of failure, the response is an error message.

Updating autoarchive rules for a job output file

Use UpdateJobSchedule to change autoarchive rules for the output of a job when you update the job schedule. For example, you can change rules about using the inherited archive policy for the document's file type, about deleting the output, or about archiving the document before deletion. The following example updates the autoarchive rules for inheritance, expiration age, and whether to archive the file at expiration:

```
<SOAP-ENV:Body>
   <Administrate>
      <UpdateJobSchedule>
         <SetAttributes>
            <RunLatestVersion>false</RunLatestVersion>
            <InputFileName>/detail.rptdesign; 2</InputFileName>
         </SetAttributes>
         <SetParameters>
            <OutputMaxVersion>0</OutputMaxVersion>
            <RetryOption>VolumeDefault</RetryOption>
            <ArchiveRuleInherited>false</ArchiveRuleInherited>
            <ExpirationAge>86400</ExpirationAge>
            <ArchiveOnExpire>true</ArchiveOnExpire>
         </SetParameters>
         <SetParameterValues>
            <ParameterValue>
               <Group>Office Parameters</Group>
               <Name>DataSource::offices_city</Name>
            </ParameterValue>
            …
         </SetParameterValues>
         <Id>13</Id>
      </UpdateJobSchedule>
   </Administrate>
</SOAP-ENV:Body>
```

As with other Administrate operations, the response to this request is empty when the operation succeeds. In the event of failure, the response is an error message.

Updating the autoarchive rules for a file type in a folder or volume

Use UpdateFile to change the default autoarchive rules for a file type in a folder or volume. Indicate the file type in the ArchiveRule element. In the NameList

element, specify the folder to which the new rules apply. In the following example, the new rules apply to all files with the .rpt extension in the Inventory folder, unless individual files of that type already have autoarchive rules. Express the expiration age in minutes.

```
<SOAP-ENV:Body>
   <Administrate>
      <UpdateFile>
         <SetArchiveRules>
            <ArchiveRule>
               <FileType>rpt</FileType>
               <NeverExpire>false</NeverExpire>
               <ArchiveOnExpiration>true</ArchiveOnExpiration>
               <ExpirationAge>2880</ExpirationAge>
            </ArchiveRule>
         </SetArchiveRules>
         <NameList>
            <String>/Inventory</String>
         </NameList>
      </UpdateFile>
   </Administrate>
</SOAP-ENV:Body>
```

As with other Administrate operations, the response to this request is empty when the operation succeeds. If the request fails, the response is an error message.

Setting an autoarchive schedule when updating an Encyclopedia volume

Use UpdateVolumeProperties and change properties of SetAutoArchiveSchedules to change an Encyclopedia volume's autoarchive schedule.

```
<SOAP-ENV:Body>
   <Administrate>
      <UpdateVolumeProperties>
         <SetAutoArchiveSchedules>
            <SetSchedules>
               <TimeZoneName>PST</TimeZoneName>
               <ScheduleDetails>
                  <JobScheduleDetail>
                     <ScheduleType>Daily</ScheduleType>
                     <ScheduleStartDate>2008-12-12
                        </ScheduleStartDate>
                     <ScheduleEndDate>2008-12-28</ScheduleEndDate>
                     <DatesExcluded>
                        <Date>2008-12-25</Date>
                     </DatesExcluded>
                     <DurationInSeconds>1800</DurationInSeconds>
```

```
                    <Daily>
                       <FrequencyInDays>1</FrequencyInDays>
                       <OnceADay>07:00:00</OnceADay>
                    </Daily>
                 </JobScheduleDetail>
              </ScheduleDetails>
           </SetSchedules>
        </SetAutoArchiveSchedules>
     </UpdateVolumeProperties>
   </Administrate>
</SOAP-ENV:Body>
```

As with other Administrate operations, the response to this request is empty when the operation succeeds. If the request fails, the response is an error message.

Starting an archive process for an Encyclopedia volume

Use ExecuteVolumeCommand and set the StartArchive command to start archiving the items in an Encyclopedia volume.

```
<SOAP-ENV:Body>
   <ExecuteVolumeCommand>
      <VolumeName>Monaco</VolumeName>
      <Command>StartArchive</Command>
   </ExecuteVolumeCommand>
</SOAP-ENV:Body>
```

The response returns a status of the command.

```
<SOAP-ENV:Body>
   <ExecuteVolumeCommandResponse>
      <Status>Succeeded</Status>
   </ExecuteVolumeCommandResponse>
</SOAP-ENV:Body>
```

Retrieving autoarchive rules for a file or folder

Use GetFileDetails to retrieve the autoarchive rules for a file or folder. Identify the file, then specify ArchiveRules as a string in ResultDef, as shown in the following example:

```
<SOAP-ENV:Body>
   <GetFileDetails>
      <FileId>4</FileId>
      <ResultDef>
         <String>ArchiveRules</String>
      </ResultDef>
```

```
   </GetFileDetails>
</SOAP-ENV:Body>
```

The response returns identifying information about the file, followed by the autoarchive rules for the file and the folder that contains it. In the following example, the file type $$$ indicates that the rule applies to a folder. The expiration age is in minutes.

```
<SOAP-ENV:Body>
   <GetFileDetailsResponse>
      <File>
         <Id>4</Id>
         <Name>/Inventory/pie.rptdesign</Name>
         <FileType>RPTDESIGN</FileType>
         <TimeStamp>2008-12-11T22:12:17</TimeStamp>
         <Owner>Administrator</Owner>
         <UserPermissions>VSRWEDG</UserPermissions>
         <Version>1</Version>
         <PageCount>12</PageCount>
         <Size>74240</Size>
      </File>
      <ArchiveRules>
         <ArchiveRule>
            <FileType>RPTDESIGN</FileType>
            <NeverExpire>false</NeverExpire>
            <ExpirationAge>23040</ExpirationAge>
            <ExpireDependentFiles>false</ExpireDependentFiles>
            <ArchiveOnExpiration>true</ArchiveOnExpiration>
            <IsInherited>false</IsInherited>
         </ArchiveRule>
         <ArchiveRule>
            <FileType>$$$</FileType>
            <NeverExpire>false</NeverExpire>
            <ExpirationAge>23040</ExpirationAge>
            <ExpireDependentFiles>false</ExpireDependentFiles>
            <ArchiveOnExpiration>true</ArchiveOnExpiration>
            <IsInherited>true</IsInherited>
            <InheritedFrom>/Inventory</InheritedFrom>
         </ArchiveRule>
      </ArchiveRules>
   </GetFileDetailsResponse>
</SOAP-ENV:Body>
```

Setting job notice expiration for all users

Use the DefaultSuccessNoticeExpiration and DefaultFailureNoticeExpiration elements of Volume to set the job notice expiration attribute for all users whose notice expiration is not set or is set to 0. You can set the expiration time for a

success notice or a failure notice when you create or update a volume. Express the expiration time in minutes.

In the following example, the volume properties are updated to set the default success notices to expire in three days and default failure notices to never expire. These expiration rules apply to all users whose respective notice expiration is not set or is set to 0.

```
<SOAP-ENV:Body>
   <Administrate>
      <UpdateVolumeProperties>
         <SetAttributes>
            <DefaultSuccessNoticeExpiration>4320
            </DefaultSuccessNoticeExpiration>
            <DefaultFailureNoticeExpiration>0
            </DefaultFailureNoticeExpiration>
         </SetAttributes>
      </UpdateVolumeProperties>
   </Administrate>
</SOAP-ENV:Body>
```

Setting job notice expiration for a user

Use the User element of CreateUser to set the user's job notice expiration attribute. You can set the expiration time for a success notice or a failure notice when you create or update a user. Express the expiration time in minutes. If you do not set the expiration time or set it to 0, the value specified for all users applies. To set the user's notices to never expire, set the value to 0xffffffff. In the following example, the user's success notices expire in three days and failure notices never expire:

```
<SOAP-ENV:Body>
   <Administrate>
      <Transaction>
         <CreateUser>
            <User>
               …
               <SuccessNoticeExpiration>4320
                  </SuccessNoticeExpiration>
               …
               <FailureNoticeExpiration>0xffffffff
                  </FailureNoticeExpiration>
            </User>
         </CreateUser>
         …
      </Transaction>
   </Administrate>
</SOAP-ENV:Body>
```

Chapter

14

Archiving APIs

This chapter provides an alphabetical list of SOAP-based archiving API operations and data types. Each entry includes a general description of the operation, its schema, and a description of its elements. The archiving API to creates an application that archives files from an Encyclopedia volume. This chapter consists of the following topics:

- SOAP-based archiving API operations
- SOAP-based archiving data types

SOAP-based archiving API operations

This section describes SOAP-based archiving operations.

DeleteExpiredFiles

Informs iHub that the files in ExpiredFileIds are archived and instructs iHub to delete those files.

iHub deletes the file only if the file is expired. iHub sends the ID of each expired file in the GetNextExpiredFiles response. If ExpiredFileIds contains an ID of a file that iHub did not send, iHub ignores it.

If there are no IDs of expired files in any DeleteExpiredFiles call, iHub keeps expired files in the Encyclopedia volume and sends them to the archive application at the next archive pass.

Request schema

```
<xsd:complexType name="DeleteExpiredFiles">
   <xsd:sequence>
      <xsd:element name="SessionID" type="xsd:string"/>
      <xsd:element name="ExpiredFileIds"
         type="typens:ArrayOfString"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

SessionID
The ID that iHub generates for the current archiving session.

ExpiredFileIds
The IDs of the files that were archived and can be deleted.

Response schema

```
<xsd:complexType name="DeleteExpiredFilesResponse"/>
```

EndArchive

Ends an archiving session. After iHub receives the first StartArchive call, it allows 24 hours between subsequent archive requests. If iHub does not receive any archive requests within this time period, it automatically terminates the archive session.

Request schema

```
<xsd:complexType name="EndArchive">
   <xsd:sequence>
      <xsd:element name="SessionID" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

SessionID
The ID that iHub generates for the current archiving session.

Response schema

```
<xsd:complexType name="EndArchiveResponse"/>
```

GetNextExpiredFiles

Retrieves information about expired files.

iHub always returns the ID, name, version, type, and location of each expired file. Use the ResultDef element to retrieve additional information.

Request schema

```
<xsd:complexType name="GetNextExpiredFiles">
   <xsd:sequence>
      <xsd:element name="SessionID" type="xsd:string"/>
      <xsd:element name="ResultDef" type="typens:ArrayOfString"
         minOccurs="0"/>
      <xsd:element name="MaxFiles" type="xsd:long"
         minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Request elements

SessionID
The ID that iHub generates for the current archiving session.

ResultDef
Requests the following information about the expired files:

- Description
  The description of the file
- PageCount
  The number of pages in the file
- Size
  The size of the file
- TimeStamp
  The time the file was created or modified, in Coordinated Universal Time (UTC)
- VersionName
  The version name of the file
- Owner
  The owner of the file

- AccessType
  The file's access type, private or shared
- ACL
  The access rights to the file
- DependOnFiles
  Information about the files on which the file depends

MaxFiles
The maximum number of files to retrieve and return in the result set. If not specified, the value is 1. iHub can return up to 500 files.

Response schema

```
<xsd:complexType name="GetNextExpiredFilesResponse">
   <xsd:sequence>
      <xsd:element name="ExpiredFiles"
         type="typens:ArrayOfFileInfo"/>
      <xsd:element name="HasMore" type="xsd:boolean"/>
   </xsd:sequence>
</xsd:complexType>
```

Response elements

ExpiredFiles
Information about the expired files.

HasMore
Indicates whether more expired files are available. If True, more expired files are available.

StartArchive

Starts an archive pass. StartArchive is the first call that the API makes after initializing. After iHub receives the command to start the archive application, iHub waits five minutes to receive the StartArchive request. If it does not receive the request, iHub ignores the command and invalidates the SessionID.

Request schema

```
<xsd:complexType name="StartArchive">
   <xsd:sequence>
      <xsd:element name="SessionID" type="xsd:string"/>
      <xsd:element name="ProviderName" type="xsd:string"
         minOccurs="0"/>
      <xsd:element name="IncludeFolder" type="xsd:boolean"
         minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Request elements

SessionID
The ID that iHub generates for the current archiving session.

ProviderName
A string identifying the archiving application.

IncludeFolder
A flag indicating whether to include subfolders.

Response schema

```
<xsd:complexType name="StartArchiveResponse"/>
```

SOAP-based archiving data types

This section describes SOAP-based archiving data types.

ArrayOfFileInfo

A complex data type that represents an array of FileInfo elements.

Schema

```
<xsd:complexType name="ArrayOfFileInfo">
   <xsd:sequence>
      <xsd:element name="FileInfo" type="typens:FileInfo"
         maxOccurs="unbounded" minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

ArrayOfPermission

A complex data type that represents an array of Permission elements.

Schema

```
<xsd:complexType name="ArrayOfPermission">
   <xsd:sequence>
      <xsd:element name="Permission" type="typens:Permission"
         maxOccurs="unbounded" minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

FileAccess

A simple data type that states whether a file is private or shared.

Schema

```
<xsd:simpleType name="FileAccess">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Private" />
      <xsd:enumeration value="Shared" />
   </xsd:restriction>
</xsd:simpleType>
```

FileInfo

A complex data type that contains file information.

Schema

```
<xsd:complexType name="FileInfo">
   <xsd:sequence>
      <xsd:element name="Id" type="xsd:string" />
      <xsd:element name="Name" type="xsd:string" />
      <xsd:element name="Version" type="xsd:long" />
      <xsd:element name="FileType" type="xsd:string" />
      <xsd:element name="FileLocation" type="xsd:string" />
      <xsd:element name="Description" type="xsd:string"
         minOccurs="0" />
      <xsd:element name="PageCount" type="xsd:long"
         minOccurs="0"/>
      <xsd:element name="Size" type="xsd:long" minOccurs="0" />
      <xsd:element name="TimeStamp" type="xsd:dateTime"
         minOccurs="0" />
      <xsd:element name="VersionName" type="xsd:string"
         minOccurs="0" />
      <xsd:element name="Owner" type="xsd:string" minOccurs="0" />
      <xsd:element name="AccessType" type="typens:FileAccess"
         minOccurs="0" />
      <xsd:element name="ACL" type="typens:ArrayOfPermission"
         minOccurs="0" />
      <xsd:element name="DependOnFiles"
         type="typens:ArrayOfFileInfo" minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

Elements

Id
The file ID.

Name
The file name.

Version
The file version number.

FileType
The file type.

FileLocation
The location of the file.

Description
A description of the file.

PageCount
The number of pages within the file.

Size
The size of the file.

TimeStamp
The timestamp on the file.

VersionName
The file version name.

Owner
The owner of the file.

AccessType
The file access type.

ACL
The access control list for the file.

DependOnFiles
A list of files that this file is dependent on.

Permission

A complex data type that describes access rights for roles or users.

Schema

```
<xsd:complexType name="Permission">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="RoleName" type="xsd:string" />
         <xsd:element name="UserName" type="xsd:string" />
      </xsd:choice>
      <xsd:element name="AccessRight" type="xsd:string" />
   </xsd:sequence>
</xsd:complexType>
```

Elements

RoleName
The role name being described.

UserName
The user name being described.

AccessRight
The privileges the user or role has on an object. One or more of the following characters representing a privilege:

- D—Delete
- E—Execute
- G—Grant
- V—Visible
- S—Secured Read
- R—Read
- W—Write

Chapter

15

Customizing installation on Windows systems

This chapter discusses the following topics:

- Extracting an Actuate product installation package
- Modifying the installed files and registry entries
- Localizing the installation
- Creating a silent installation
- Performing a silent installation
- Performing a silent installation removal
- Performing a silent removal of Actuate Localization and Online Documentation

Extracting an Actuate product installation package

Actuate no longer distributes its software products on DVD installation media. A customer downloads an Actuate product installation package from an FTP software distribution site.

In the Windows environment, Actuate distributes the product installation package in a self-extracting executable archive file, such as an EXE file. An administrator can customize the Actuate installation program for a specific environment to localize or create a silent installation.

To customize an Actuate product installation, the administrator first extracts the installation files from the archive file to a local machine. After extracting the contents of the installation package, the administrator can use a text editor to customize the configuration files.

Modifying the installed files and registry entries

You can customize the appearance of the Actuate product installation on Windows by modifying the file, custom.ini, to make the following changes to the Actuate installation:

- Change the paths to files that install with an Actuate product.
- Install custom files during the Actuate installation.
- Change registry key settings during the Actuate installation.

Use caution when you edit Windows registry keys. Changes to registry key entries and values can affect the Windows operating system.

The custom.ini file is located in the \custom directory of the Actuate product installation files. After you modify custom.ini, create new installation media that contains custom.ini and other custom files.

How to modify custom.ini

1. For example, download the iHub distribution package from an FTP software distribution site. Using WinZip or other archiving utility, extract the contents of the self-extracting executable file, ActuateBIRTiHub.exe, to a directory on the local machine, such as C:\Actuate\iHub.
2. In a text editor, open custom.ini in the Actuate product directory.

   For all products, the custom.ini contains code that lists an external DLL and registry file, similar to the following excerpt:

```
; Modify the following entries to allow custom installation
     of files and/or registry entries. Uncomment the
     entries to support this feature.

[AC_EXTERNAL_FILES]
;FILE1=c:\tmp\abc.dll
;FILE2=d:\tmp\abc.pdf

[AC_REGISTRY_ENTRY]
;RegistryFile1=aaaa.reg
```

3 To add files to the installation, perform the following steps:

1. Uncomment code in the [AC_EXTERNAL_FILES] list by deleting the semicolon (;) that precedes the code.
2. Edit the code to include the files that you want to install. You must use full file paths to specify the directories into which to install files.

   For example, type:

   ```
   FILE1=C:\WINDOWS\system32\My_dll.dll
   FILE2=C:\Program Files\Actuate\iHub\Help
      \My_help_file.pdf
   FILE<n>=C:\Program Files\Actuate\iHub\Help
      \My_help_file.html
   ```

   In this example, <n> is the total number of files. You can add files up to the limit of the InstallShield version.
3. Copy the files that you added in the paths in substep 2 to the \custom subdirectory.

4 To add custom registry entries with the installation, perform the following steps:

1. Uncomment code in the [AC_REGISTRY_ENTRY] list by deleting the semicolon (;) that precedes the code.
2. Edit the code to include the registry file that contains the custom registry entries. For example, type:

   ```
   RegistryFile1=Customer.reg
   ```

   where Customer.reg is the registry file. A Customer.reg entry looks like the following entry:

   ```
   [HKEY_LOCAL_MACHINE\SOFTWARE\MyProduct\TestEntry\Custom Key]
   "MyApp"="C:\\MyDirectory\\MyApplication"
   "AppVersion"="8.0.0.0"
   "Value"=dword:00000002
   ```

In this example, double quotation marks enclose strings and string values. A numeric value appears in the following example:

```
"Value"=dword:00000002
```

3. Copy the registry file to the \custom subdirectory.

5. Save custom.ini.
6. Test the installation by running the setup.exe file in the product installation directory.

Localizing the installation

Actuate uses InstallShield and assumes that you are familiar with this program. You can change the installation appearance to suit regional preferences by using locale-specific installation libraries provided by Actuate or by directly customizing the InstallShield installation.

To localize an Actuate installation, you can use InstallShield in the following ways:

- Change the Setup.gif image that appears at the start of the installation.
- Change the application name that appears during the installation by modifying setup.ini.
- Change the text of the resource IDs in brand.rc for an Actuate product, and rebuild ac_brand.dll to change the content of the Actuate dialog boxes that appear during the installation.
- Change custom InstallShield dialog boxes by extracting _isuser.dll from a cabinet file and modifying _isuser.dll.

After you change these files, test the installation process, and make the installation media.

The following procedures refer to the setup files contained in the Actuate product distribution package. Download the distribution package from the FTP or other site. Using WinZip or other archiving utility, extract the contents of the self-extracting executable file to a local machine. Modify the standard installation files to create a custom installation.

How to change the image that appears during installation

To replace the image that appears in the dialog box at the start of the installation, replace setup.gif (Graphics Interchange Format) in the Actuate product installation with another .gif file. This image is also called a splash screen. The file name must be setup.gif for InstallShield to find and use the image.

How to change values that appear during installation

Edit the setup.ini file to change the application name, the company name, the number of seconds the splash screen appears, and other values. The following steps explain how to change the application name:

1. Using a text editor, open setup.ini.

   In setup.ini, the key<n> entries, such as key0=0x0804, refer to the ac_brand files that Actuate supplies. Table 15-1 lists the files. Not all Actuate products support all locales.

   Table 15-1 ac_brand files supplied by Actuate

Locale	File name	Code
Chinese	ac_brand.chs	0x0804
English	ac_brand.dll	0x0009
French	ac_brand.fra	0x040c
German	ac_brand.deu	0x0007
Japanese	ac_brand.jpn	0x0011
Korean	ac_brand.kor	0x0012
Spanish	ac_brand.esp	0x000a

2. Change the value of AppName to the desired name.
3. Save setup.ini.

How to change the text of the Actuate installation dialog boxes

This procedure uses Microsoft Visual C++ version 6.0 to customize the text in the Actuate dialog boxes that appear during installation. You can customize this text to localize the Actuate installation dialog boxes for languages other than US English. For more information about using Visual C++, see the Microsoft Visual C++ documentation.

To customize or localize InstallShield custom dialog boxes, you must modify a DLL file that is in the Actuate product installation.

1. Using Microsoft Visual C++, open the ac_brand.dll in the Actuate product installation as Resources in an executable file, as shown in Figure 15-1.

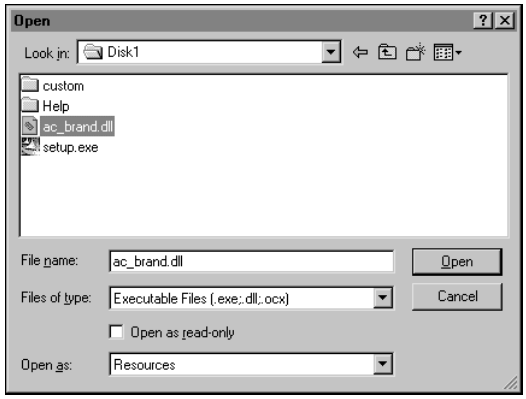


Figure 15-1 Customizing InstallShield dialog boxes

2 In ac_brand.dll, expand the following items:

- Bitmap
- String Table
- Version

The expanded view appears, as shown in Figure 15-2.

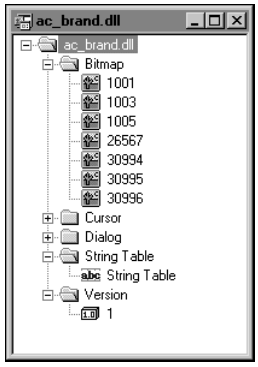


Figure 15-2 ac_brand.dll, expanded

3 Right-click a resource, such as a bitmap, and choose Properties.

4 In Properties, you can change the following settings:

- For Bitmap, change the Language, Condition, and File name.
- For String Table, change the Language.
- For Version, change Language and Condition.

5 To modify the text for a dialog box, perform the following steps:

1 In the expanded view, double-click String Table.

A list of IDs, values, and captions appears in the right pane, as shown in Table 15-2.

Table 15-2 Sample dialog box IDs, values, and captions

ID	Value	Caption
IDS_MSG_COPYING	13004	Copying files to the computer.
IDS_SETUP_FINISH_MSG	13019	Setup completed successfully.

2 Double-click a caption to modify the text.

3 In String Properties, modify the caption text.

To save the change, close String Properties.

4 Repeat steps 2 and 3 for all dialog box captions.

6 Save ac_brand.dll.

How to change InstallShield's custom dialog boxes

The following procedure uses the InstallShield utility, iscab.exe, to extract the internal DLL, _isuser.dll. To customize the text in the InstallShield custom dialog boxes, modify _isuser.dll using Visual C++. The _isuser.dll is in the data1.cab cabinet file. For more information about using iscab.exe, refer to the InstallShield documentation.

1 Using Windows Explorer, copy data1.cab from the Actuate product installation to the customization directory on the local machine.

2 Copy iscab.exe to the directory that contains data1.cab.

3 Create a configuration file for the InstallShield IsCab utility.

1 Create a text file named file.ini in the directory that contains data1.cab.

2 Type the following lines into the file:

```
[ISCAB Info]
Product=ISCAB
Version=2.0

[<Support>Language Independent OS Independent Files]
File1="_Isuser.dll"
```

3 Save and close the file.

4 Choose Start➛Programs➛Accessories➛Command Prompt.

5. In Command Prompt, change directories to the directory that contains data1.cab.
6. To extract _isuser.dll from data1.cab, type the following command, then press Enter:

   ```
   iscab data1.cab -x -ifile.ini
   ```

   iscab.exe extracts _isuser.dll to the local directory.
7. Using Visual C++, modify _isuser.dll to customize the dialog boxes. Do not change ID names or values in the file.
8. To remove the original _isuser.dll file from the cabinet file, type the following command at the command prompt, then press Enter:

   ```
   iscab data1.cab -r -ifile.ini
   ```
9. To add the modified _isuser.dll to the cabinet file, type the following command, then press Enter:

   ```
   iscab data1.cab -a -ifile.ini
   ```

Creating a silent installation

You can customize the installation so that InstallShield installs and uninstalls Actuate products silently, without user interaction. By specifying which dialog boxes appear, you can create a completely or partially silent installation. You can also collect any messages or other output in a log file. During a completely silent installation, the splash screen and input dialog boxes do not appear.

To create a silent installation, provide an XML file that drives the silent installation for the Actuate product. This file contains parameters to control the installation. You can install the following Actuate products silently:

- BIRT Designer Professional
- BIRT iHub
- BIRT iHub Integration Technology
- Information Console
- Localization and Online Documentation

The silent installation uses the following files listed in Table 15-3. Actuate provides these files in the Actuate product installation directory.

Table 15-3 Silent installation files

File name	Description
acis.iss	The InstallShield response file. Do not modify this file.

Table 15-3 Silent installation files

File name	Description
acinstallinput.xml	The silent installation sample input. The default version of this file creates a completely silent installation. Customize this file to match the installation requirements.
acinstallinput.xsd	The silent installation schema. Use this file with an XML development tool to edit the silent installation file.

You can customize the silent installation of an Actuate product by editing the acinstallinput.xml file.

You cannot perform a silent rollback of an upgrade installation for BIRT iHub or Information Console.

The root element in acinstallinput.xml is Config. Config has three child elements, as described in Table 15-4.

Table 15-4 Config child elements

Element	Description
VersionInfo	Information about the software version. For more information about VersionInfo, see "Specifying version information," later in this chapter.
GeneralDlgs	All dialog boxes that appear during a typical installation. For more information about GeneralDlgs, see "Customizing installation dialog boxes," later in this chapter.
CustomDlgs	All dialog boxes that appear only in a custom installation. For more information about CustomDlgs, see "Customizing installation dialog boxes," later in this chapter.

How to modify the silent installation file

The following procedure uses a text editor to modify the acinstallinput.xml file for BIRT iHub as an example. The modifications create a partially silent custom installation. Modify the acinstallinput.xml file for other products in a similar way.

1. Download the iHub distribution package from an FTP software distribution site. Using WinZip or other archiving utility, extract the contents of the self-extracting executable file, ActuateBIRTiHub.exe, to a directory on the local machine, such as C:\Actuate\iHub.
2. Save a copy of acinstallinput.xml as orig_acinstallinput.xml so that you have a copy of the original silent installation input file.
3. Using a text editor that can handle UTF-8 encoding, open acinstallinput.xml.

4 Search for WelcomeDlg. Change its Visible value to True, as shown in the following example:

```
<WelcomeDlg Visible="true"/>
```

This causes the Welcome dialog box to appear during the installation. Because all dialog boxes have their Visible element set to False by default, only the Welcome dialog box appears during the installation.

5 Search for SetupTypeCtl. Change its value to Custom, as shown in the following example:

```
<SetupTypeCtl>Custom</SetupTypeCtl>
```

This change specifies a custom installation. The XML Schema Document, acinstallinput.xsd, requires either Typical and Custom as the only valid values for SetupTypeCtl.

6 Search for ComponentsDlg. Change the value of the Examples element to False, as shown in the following example:

```
<Examples>false</Examples>
```

This change means that the installation does not install the Example files.

7 Search for ProgramFolderCtl. Change Actuate 11 to Actuate 11 iHub, as shown in the following example:

```
<ProgramFolderCtl>Actuate 11 iHub </ProgramFolderCtl>
```

This change places iHub in the Start menu as Start➛Programs➛Actuate 11 iHub.

8 Save and close acinstallinput.xml.

9 Perform the steps in "How to use acinstallinput.xml to perform a silent installation," later in this chapter, to test the modified acinstallinput.xml file.

During the installation, the only dialog box that appears is the Welcome dialog box. After the installation, use Windows Explorer to view the contents of C:\Program Files\Actuate\iHub. There is no Examples subdirectory. Also verify that the Start➛Programs➛Actuate 11 iHub menu item exists.

Specifying version information

VersionInfo is an optional element with two optional child elements, Version and Copyright. The default acinstallinput.xml file contains version and copyright information, as shown in the following example:

```
<VersionInfo>
   <Version>11 Service Pack 4 (Build DEV120911)</Version>
   <Copyright>Copyright @1995-2013 Actuate Corporation
      </Copyright>
</VersionInfo>
```

The installation checks the version only if <VersionInfo><Version> exists and is not empty. If the version checking fails, the installation procedure writes a message to the log file and the installation continues.

Specifying license information

Only the acinstallinput.xml file for BIRT iHub includes license information. BIRT iHub requires a valid license file to install and enable the purchased options.

To specify the license file location, set the <LicenseFileCtl> element of <LicenseFileDlg>. The default license file location is empty. Add the license file location as shown in the following example:

```
<LicenseFileCtl>C:\Temp\ServerLicense.xml</LicenseFileCtl>
```

Customizing installation dialog boxes

This section describes how to customize typical and custom installation dialog boxes. It is only necessary to customize the dialog boxes used by a custom installation.

Both typical and custom installations require the GeneralDlgs of acinstallinput.xml. Each dialog box element in GeneralDlgs corresponds to a dialog box in the installation user interface. For example, SetupTypeDlg is the dialog box that asks if this will be a typical or custom installation and where to install the software.

Only a custom installation uses the CustomDlgs of acinstallinput.xml. Each dialog box element in CustomDlgs corresponds to a dialog box that appears only during a custom installation. For example, ComponentsDlg is the dialog box that asks which product components to install. If the silent installation is a typical install, then you can omit the CustomDlgs component.

Specifying dialog box information

Each dialog box element has a Visible attribute and an unlimited number of control child elements. For instance, SetupTypeDlg has SetupTypeCtl and LocationCtl child elements that correspond to the setup type control and installation location control in the dialog box. Set the Visible attribute to False to prevent the display of a dialog box. Include the necessary control elements and values to supply the dialog box data. For a completely silent installation, set all Visible attributes to False. For a partially silent installation, set Visible to True for only those dialog boxes that require user interaction.

Each Actuate product uses its own set of dialog boxes. Examine the default acinstallinput.xml and acinstallinput.xsd files for a product to see the available elements and controls and view their default values.

Encrypting dialog box information

You can install BIRT iHub to require a password to log in to the Configuration Console. The default BIRT iHub acinstallinput.xml file contains the following dialog box definition for the system administrator account:

```
<ConfigActuateSystemAdminDlg Visible="false">
   <PasswordCtl Password="" Encrypt="false">
   </PasswordCtl>
</ConfigActuateSystemAdminDlg>
```

There is no default password. To add a default password in clear text, replace the empty Password string with a password, as shown in the following example:

```
<PasswordCtl Password="ThePassword" Encrypt="false">
```

In this example, the string ThePassword is the default password.

To encrypt the password, use the Actuate acencrypt utility to first encode the password before you place it in acinstallinput.xml. Actuate provides acencrypt.exe in the Actuate product installation directory.

Using acencrypt

The acencrypt command-line utility converts the first line of text in a file to an encrypted line in an output file. The acencrypt utility accepts ASCII and non-ASCII strings.

Table 15-5 lists the acencrypt parameters.

Table 15-5 acencrypt parameters

Parameter	Description
-input <inputFile>	Required parameter specifies the file to encrypt. Only the first line is processed as the password and subsequent entries are ignored.
-output <outputFile>	Optional parameter specifies the encrypted output file. The default is to display the encrypted string on standard out.

To use an encrypted password, use acencrypt to encrypt the password, copy and paste the encrypted password in acinstallinput.xml, and set the Encrypt attribute to True. Using the value of True for the Encrypt attribute means the installation program decrypts the password before using it. Otherwise, use the default value, False, and provide the password as plain text.

How to encrypt a string with acencrypt

The following procedure encrypts a clear text password, ThePassword, and adds the encrypted text to an installation file:

1. Using a text editor, create clear.txt with ThePassword as its only line.

   ```
   ThePassword
   ```

2. Save and close clear.txt.
3. Choose Start➛Programs➛Accessories➛Command Prompt.

   Command Prompt appears.

4. At the command prompt, type the following command, then press Enter:

   ```
   acencrypt -input clear.txt -output encrypted.txt
   ```

   The acencrypt utility creates the file encrypted.txt.

5. Using a text editor, open encrypted.txt. It contains the following line:

   ```
   24262b27292c23242e2625272e292e2a2c2828262d25
   ```

6. Using the editor, open acinstallinput.xml for the Actuate installation you are customizing.
7. In acinstallinput.xml, copy and paste the encrypted string from encrypted.txt into the dialog box's password control and set Encrypt to True, as shown in the following example:

   ```
   <PasswordCtl
      Password="24262b27292c23242e2625272e292e2a2c2828262d25"
      Encrypt="true">
   ```

Performing a silent installation

You use the default acinstallinput.xml file to silently install an Actuate product. Before acinstallinput.xml can be used to perform the silent installation, you have to provide the following minimal information:

- <AgreementCtl> must be set to yes.
- <LicenseFileTypeCtl> must be set to trial, or a path to the license file must be provided.
- In <ServiceProfile> the <usernameCtl> and <PasswordCtl> must be provided with the appropriate user name and password information.
- The typical installation uses PostgreSQL database. A password must be provided for <PostgreSQLServiceProfile> and <PostgreSQLDatabaseProfile> through <PasswordCtl> in acinstallinput.xml.

The following procedure illustrates how to perform a silent installation.

How to use acinstallinput.xml to perform a silent installation

This procedure uses the provided acinstallinput.xml for a default silent installation configuration. For information about creating a custom silent installation file, see "How to modify the silent installation file," earlier in this chapter.

1. Using Windows Explorer, copy the Actuate product installation directory to the local machine.

   For example, for BIRT iHub, copy the installation files from the iHub installation directory to C:\Actuate\iHub.

2. Choose Start➛Programs➛Accessories➛Command Prompt.

   Command Prompt appears.

3. At the command prompt, go to the product installation directory that you created on the local machine. For example, type the following command, then press Enter:

   ```
   cd C:\Actuate\iHub
   ```

4. Invoke the silent installation in asynchronous or synchronous mode.

   - For asynchronous silent installation, type the following command, then press Enter:

     ```
     setup.exe -s -f1"C:\Actuate\iHub\acis.iss"
        -acinput "C:\Actuate\iHub\acinstallinput.xml"
        -acoutput "C:\Actuate\iHub\iHub_log.xml"
     ```

     The command prompt reappears, and the installation completes in the background.

   - For synchronous silent installation, type the following command, then press Enter:

     ```
     start /wait setup.exe -s -f1"C:\Actuate\iHub\acis.iss"
        -acinput "C:\Actuate\iHub\acinstallinput.xml"
        -acoutput "C:\Actuate\iHub\iHub_log.xml"
     ```

     The command prompt reappears after the installation completes.

You must fully specify the files for the -f1, -acinput, and -acoutput options. There is no space between -f1 and the full acis.iss file name. There is a space between the -acinput and -acoutput options and their file names. The -acoutput file name may contain only ASCII characters. If you do not specify an -acoutput file, the installation generates the default log file, acinstparam.xml, in the installation destination folder. The XML log file for the silent installation appears in the directory where acinstallinput.xml and the Actuate product installation executable file, setup.exe, reside.

In a default acinstallinput.xml, there are two destination folders for the installation, the Binary location and the Data location. iHub uses the Binary location to resolve paths to all the binaries that it launches. The default path for the Binary location is C:\Program Files\Actuate\iHub, and is referred to in the iHub documentation as the environment variable AC_SERVER_HOME. iHub uses the Data location to store iHub logs, iHub Encyclopedia including PostgreSQL data, MC logs, IC logs, and all other data. The default path for the Data location is C:\Actuate\iHub\data, and is referred to in the iHub documentation as the environment variable AC_DATA_HOME.

How to verify a successful silent installation

You verify a silent installation on a Windows server by examining the installation log. The -acoutput option creates a log in the directory where acinstallinput.xml and the Actuate product installation executable, setup.exe, reside.

1. In a text editor, open the log created by the -acoutput option.

   The file contains the following information after a successful silent installation:

```
<Config>
   <Result>success</Result>
   <Reason/>
   <RebootDlg>
      <RebootRequired>FALSE</RebootRequired>
   </RebootDlg>
   <MessageBox>Setup completed successfully.</MessageBox>
</Config>
```

2. Look at the RebootRequired parameter value to determine whether the installation requires that you reboot. True means you must reboot the machine to finish the installation.
3. Verify that the value for Result is success. This result indicates a successful installation.

   If the Result value is not success, read the Reason parameter value to determine the nature of the problem.

Performing a silent installation removal

Execute acuninstall.exe from the command line to remove an installation. The syntax is:

```
acuninstall -a -p acuninst.txt
```

The acuninstall utility has one option, the RemoveAll InstallShield option. The following example silently uninstalls a standard installation of BIRT iHub from the local machine:

```
C:\Windows\System32\acuninstall -a
   -p "C:\Program Files\Actuate\iHub\AcUninst.txt"
```

For each Actuate product, the silent installation creates a log file, acuninst.txt, in the product's top-level directory during installation. For example, the BIRT iHub acuninst.txt file is:

```
C:\Program Files\Actuate\iHub\AcUninst.txt
```

Performing a silent removal of Actuate Localization and Online Documentation

For Actuate Release 11, you install Actuate localization and online documentation files after installing Actuate products. To perform a silent removal of the Actuate Localization and Online Documentation, perform the following tasks:

- Modify the configuration file, acinstallinput.xml, for the Actuate Localization and Online Documentation.
- Perform a silent installation using the modified XML file.

In the acinstallinput.xml file for the Actuate Localization and Online Documentation, the default value of the MaintenanceTypeCtl element is MODIFY.

```
<MaintenanceTypeCtl>MODIFY</MaintenanceTypeCtl>
```

Change the value to REMOVEALL.

```
<MaintenanceTypeCtl>REMOVEALL</MaintenanceTypeCtl>
```

Chapter

16

Customizing installation on UNIX and Linux systems

This chapter discusses the following topics:

- Extracting an Actuate product installation package
- Creating a silent installation
- Modifying the parameter template file
- Performing a silent installation
- Performing a silent installation removal

Extracting an Actuate product installation package

Actuate no longer distributes its software products on DVD installation media. A customer downloads an Actuate product installation package from an FTP software distribution site.

In the Linux or UNIX environments, Actuate distributes the product installation package in an archive file, such as a TAR file. An administrator can customize the Actuate installation program for a specific environment to localize or create a silent installation.

To customize an Actuate product installation, the administrator first extracts the installation files from the archive file to a local machine. After extracting the contents of the installation package, the administrator can use a text editor to customize the configuration files.

To customize an Actuate product installation, perform the following tasks:

- Download the distribution package from the FTP or other site.
- Extract the files.
- Modify script prompts and environment variables in the shell script files.

Table 16-1 lists shell script file names used in the installations of BIRT iHub and BIRT iHub Integration Technology.

Table 16-1 Shell script file names for Actuate product installations

Actuate product	Files that contain most of the textual prompts	Files that contain most of the programming logic
BIRT iHub	isconst.sh	isinstall.sh
BIRT iHub Integration Technology	isitconst.sh	isitinstall.sh

Creating a silent installation

This section covers the following topics:

- Modifying the parameter template file
- Performing a silent installation
- Performing a silent installation removal

The following procedures describe how to create a silent installation. A silent installation requires no user interaction after the installation starts. You cannot

perform a silent rollback of an upgrade installation for BIRT iHub. You modify the template file that contains all input parameters for the installation. This file contains all user information the installation shell script requires. The entries in the template are default values for an installation. Change the default values to configure the silent installation. During the installation, the installation shell script file reads the parameter file. The parameter template files and the installation shell script files are located at the root level of the Actuate product installation directory. The names for the parameter template files and the installation shell script files for BIRT iHub and BIRT iHub Integration Technology appear in Table 16-2.

Table 16-2 Parameter template and installation script names

Actuate product	Parameter template file	Install shell script file
BIRT iHub	isinstall.cfg	isinstall.sh
BIRT iHub Integration Technology	isitinstall.cfg	isitinstall.sh

Modifying the parameter template file

You modify the parameter template file for an Actuate product to enable silent installations. In the Actuate product installation directory, the parameter template file is located in the same directory as the product's installation script file. The parameter template file contains all the required variables for a silent installation.

About isinstall.cfg

The parameter template file, isinstall.cfg, ships in the product installation for BIRT iHub. The file isinstall.cfg contains all parameters required by the installation shell script file for BIRT iHub installation. Comments in the code provide information about the parameters in each section.

Modifying isinstall.cfg

The following procedure uses the BIRT iHub parameter template file, isinstall.cfg, as an example.

How to modify isinstall.cfg

1. Log in as root.
2. Download the iHub distribution package from the FTP or other site. Extract the files.
3. Save the parameter template file, isinstall.cfg, as orig_isinstall.cfg so that you have a copy of the original parameter template file.

4 In a text editor, open the parameter template file, isinstall.cfg.

5 Edit the values for variables. Comments in the code provide information about the variables and appropriate values.

For the installation information for LDAP and database drivers sections in the code, setting the first variable value to n means the installation does not use the additional values in the section. In the following example, isinstall.cfg contains code that specifies the installation does not include settings for LDAP:

```
# Installation information for LDAP

S_AC_RS_INTEGRATE_LDAP=n
S_AC_RS_LDAP_MACHINE=build1-sun
S_AC_RS_LDAP_PORT=2222
```

Because the value for S_AC_RS_INTEGRATE_LDAP is n, the installation does not use the values for the machine name and port number.

6 Save the modified parameter file.

7 Include the modified parameter file as the installation shell script file in the new installation directory.

Performing a silent installation

The silent installation for BIRT iHub and BIRT iHub Integration Technology uses command line entries and creates log files that appear in Table 16-3.

Table 16-3 Silent installation command line entries and log files

Actuate product	Command line entry	Installation log file
BIRT iHub	./isinstall.sh -s isinstall.cfg	/tmp/$LOGNAME/isinstall.log
BIRT iHub Integration Technology	./isitinstall.sh -s isitinstall.cfg	/tmp/$LOGNAME/isitinstall.log

A new log file is created every time a user installs an Actuate product. The log files are distinguished by including the user's login name ($LOGNAME) in the log file's path.

How to perform a silent installation

The following procedure uses the silent installation for BIRT iHub as an example.

1 Log in using the account created for installing and running iHub.

2 Copy the product installation files to the local machine.

3 Change your working directory to the root directory of the product installation files.

4 To start the silent installation, type the following, then press Enter:

```
./isinstall.sh -s isinstall.cfg
```

The silent installation completes and creates the installation log file, /tmp/isinstall.log.

In a default acinstallinput.xml there are two destination folders for the installation, the Binary location and the Data location. iHub uses the Binary location to resolve paths to all the binaries that it launches. The default path for the Binary location is $HOME/AcServer, and is referred to in the iHub documentation by the environment variable AC_SERVER_HOME. iHub uses the Data location to store data, including Encyclopedia volume data, iHub logs, PostgreSQL data, MC logs, and IC logs. The default path is AC_SERVER_HOME/data, and is referred to in the iHub documentation by the environment variable AC_DATA_HOME.

Performing a silent installation removal

The silent uninstall with the -s option removes all files and directories added during installation of the Actuate product. Only the uninstall log file remains.

The silent uninstall for BIRT iHub and BIRT iHub Integration Technology uses command line entries and creates log files listed in Table 16-4.

Table 16-4 Silent uninstall command line entries and log files

Actuate product	Command line entry	Uninstallation log file
BIRT iHub	./isuninstall.sh -s	/tmp/$LOGNAME/isuninstall.log
BIRT iHub Integration Technology	./isituninstall.sh -s	/tmp/$LOGNAME/isituninstall.log

How to remove a silent installation

The following procedure shows how to perform a silent uninstall for BIRT iHub:

1 Navigate to AC_SERVER_HOME/bin, which is the directory containing isuninstall.sh.

2 To start the silent uninstall script, type the following command, then press Enter:

```
./isuninstall.sh -s
```

The command removes all BIRT iHub directories and files added during the installation, and creates /tmp/isuninstall.log.

Appendix

A

Text string limits in Actuate operations

Actuate's internal data store imposes a fixed upper limit on the length of certain text strings. An application that uses elements such as user names, URLs, file types, and descriptions must adhere to these limits. Table A-1 lists the maximum field lengths for text elements in BIRT iHub Information and Management Consoles and for elements you create using Actuate Information Delivery API.

Table A-1 Text string limits in Actuate operations

Complex data type	Element name	Maximum length, in characters
ArchiveRule	FileType	20
Attachment	ContentEncoding	10
Channel	Name	50
	Description	500
	SmallImageURL	100
	LargeImageURL	100
File	Name	255
	FileType	20
	Description	500
	VersionName	100

(continues)

Table A-1 Text string limits in Actuate operations (continued)

Complex data type	Element name	Maximum length, in characters
FileType	Name	20
	DriverName	100
	MutexClass	50
	ShortDescription	40
	LongDescription	60
	LocalExtension	20
	OutputType	20
	SmallImageURL	100
	LargeImageURL	100
	ContentType	200
JobProperties	JobName	100
	InputFileName	276
	RequestedOutputFileName	1000
	ActualOutputFileName	1000
	RequestedHeadline	100
	ActualHeadline	100
JobNotice	JobName	100
	OutputFileName	276
	OutputFileVersionName	100
	Headline	100
JobPrinterOptions	PageRange	20
	PrintToFile	256
Group	Name	50
	Description	500
Printer	Name	50
	Manufacturer (UNIX only)	50
	Model (UNIX only)	50
	Location (UNIX only)	50
	Description	100
	Orientation	20
	PageSize	50

Table A-1 Text string limits in Actuate operations (continued)

Complex data type	Element name	Maximum length, in characters
Printer *(continued)*	Resolution	20
	PaperTray	50
	Duplex	20
Role	Name	50
	Description	500
JobSchedule	TimeZoneName	32
User	Name	256
	Password	256
	EmailAddress	80
	DefaultPrinterName	50
	Description	100

Index

Symbols

, (comma) character 542
; (semicolon) character 54
? wildcard 183, 228
" (double quotation mark) character
  search results and 83, 342, 350, 507
  SOAPAction directive and 17
[] (brackets) characters 416
* (asterisk) character 16
* wildcard 98, 183, 228
\ (backslash) character 41, 416
wildcard 183, 228
= (equal sign) character 542
$$$ (file type) value 125, 611

A

ABInfoObject element 428
ABInfoObject value 484
AbsoluteDate data type 408
AbsoluteDate value 469
AC_DATA_HOME variable 635, 641
AC_EXTERNAL_FILES registry key 623
AC_JRE_HOME variable 575
AC_KEEP_WORKSPACE_DIRECTORY parameter 113
AC_SERVER_HOME variable 635, 641
AcAdminEvent table 580
AcApplicationEvent table 581
Accept directive 16
AcceptEncoding element 286, 522
access control lists
  applying 114, 115, 375, 478
  archiving 599
  changing 545
  creating 542
  deploying external 540
  getting information about 98, 101
  getting templates for 103, 294
  installing sample application for 541
  matching names or roles in 599
  replacing 114
  retrieving channel 102, 275
  retrieving external 541, 550
  retrieving file or folder 101, 104, 292, 296
  updating 116
access restrictions 98
access right attributes 488
access rights
  See also privileges
  getting 101, 104, 296
  retrieving ACL templates for 103
  setting 115, 478
  uploading files and 404
access type attributes 436
access types 437
accessing
  Apache Axis clients 162
  Encyclopedia volumes 98
  executable files 216
  iHub 26
  Online Archive Driver 598
  plug-ins 16
  PostgreSQL database 635, 641
  proxy objects 14
  sample applications 530
  third-party code libraries 163
  web services 10, 14
  WSDL schemas 11
AccessRight element
  GrantPermissions suboperations 115
  Information Deliverry API 488
  Java RSSE operations 558
  Permission data type 620
AccessRights element 493
AccessType element
  File data type 436
  FileInfo data type 619
  FileSearch data type 440
  NewFile data type 105, 478
  SelectFiles operations 118
  UploadFile operations 105
AccessType property 296
AcCreateUser class 179, 226
acDouble data type 408
AcDownloadFile_Chunked class 195, 236

acencrypt utility 632
AcErrorEvent table 582
AcErrorLogOffset table 583
AcEvent table 583
AcEventType table 584
AcExecuteReport class 199
AcExecuteReport sample application 199
acextern utility 538
AcFileType table 585
–acinput command line option 634
AcIsThreadSafe function 567, 568, 592, 593
AcJobType table 585
ACL element
  FileInfo data type 619
  GetChannelACL operations 277
  GetFileACL operations 294
  GetFileCreationACL operations 295
  GetFileDetails operations 297
  GetUserACL operations 550
  NewFile data type 478
ACL property 296
AcLogError function 568, 593
AcLogin class 172, 222, 225
AcLogUsage function 567, 592
ACLs. *See* access control lists
aclSample package 531
AcMail exceptions 574
acnotification.xml 61
acNull data type 408
AcObjectOperation table 586
AcObjectType table 586
aconlinearchive.bat 602
aconlinearchive.jar 602
aconlinearchive.sh 602
aconlinearchiveDEP.jar 602
–acoutput command line option 634
AcOutputFormat table 586
AcPerfMonExt.dll 590
AcResourceGroup table 587
acrsse directory 532
AcRSSEPassThrough function 42, 244
ACS. *See* Caching service
AcSelectFiles class 228
AcSelectJavaReportPage class 200, 202
acserverconfig.xml 539
acserverlicense.xml 539
AcServiceType table 587
AcStartErrorLog function 568, 593
AcStartUsageLog function 567, 592
AcStatus table 588
AcStopErrorLog function 568, 594
AcStopUsageLog function 567, 593
AcSystemComponent table 588
action attributes (Ping) 334
Action element 147, 334
Activate element 497, 509
Activate property 361, 392
Active Directory servers 41, 530
ActualHeadline element 467
ActualOutputFileId element 458, 466, 473
ActualOutputFileName element 458, 466, 473
ActualOutputFileSize element 458
Actuate API service 169
Actuate Basic methods 280
Actuate Basic reports 343
Actuate namespace 223
ActuateAPI class 221, 224
ActuateAPI interface 169–170
ActuateAPI service 10, 11
ActuateAPIEx interface 174, 175
ActuateAPIEx objects 234, 236
ActuateAPILocator class 169
ActuateAPILocatorEx class 175
ActuateAPILocatorEx objects 174
ActuateBuildNumber element 524
ActuateControl class 174
ActuateLog schema 580, 589
ActuateSoapPort interface 168
ActuateSoapPort_address attribute 169
ActuateVersion element 524
acuninstall utility 635
AcUploadFile class 191, 234
AcUsageLogOffset table 588
ad hoc parameters 52, 113, 143, 483
AddArchiveRules element 380
AddArchiveRules operations 605
AddChannelNotificationById element 390
AddChannelNotificationByName element 389
AddChildRolesById element 396
AddChildRolesByName element 395
AddDependentFilesById element 380
AddDependentFilesByName element 379

AddFileCreationPermissions element 401
AddGroupNotificationById element 389
AddGroupNotificationByName element 389
adding
  applications to Start menu 630
  archiving rules 598, 603, 604
  e-mail attachments 56
  file types 250, 441, 643
  folders 125, 250
  users to Encyclopedia 125, 179, 188, 226, 254
  users to notification groups 385, 388
AddLicenseOptions element 401
AddOutputFilePermissions element 390
addParameter method 192
AddParentRolesById element 396
AddParentRolesByName element 395
AddRequiredFilesById element 380
AddRequiredFilesByName element 379
address (SOAP port) 169
addressing e-mail notifications 57, 125
AddSubscribersById element 375
AddSubscribersByName element 375
AddToGroupsById element 400
AddToGroupsByName element 400
AddUserNotificationById element 389
AddUserNotificationByName element 388
addUsers method 188, 231
AddUsersById element 385
AddUsersByName element 385
Admin event type 569, 571
Administrate element 124
Administrate method 182
Administrate objects 181, 182
Administrate operations
  copying Encyclopedia items and 132
  creating folders and 125
  creating security roles and 126
  creating users and 125, 254
  defining batch operations and 187, 230, 231
  defining SOAP responses for 124
  defining transaction operations and 187, 188, 230, 231
  deleting Encyclopedia items and 127
  deleting users and 128
  described 241
  grouping transactions in 30
  handling errors with 124, 182
  managing Encyclopedia items and 29, 31
  moving file or folders and 131
  running composite 133, 135, 188, 189, 231, 232
  running single 181
  updating Encyclopedia and 128, 129, 130
Administrate requests 181, 187, 227, 231
Administrate responses 181, 182, 189
Administrate type definition 241
AdministrateResponse type definition 241
administration applications 178, 226
administration operation classes 187, 231
administration operations
  See also Administrate operations; AdminOperation operations
  developing 178, 181, 226, 241
  running 181
  submitting requests for
    Apache Axis clients 178, 181, 182, 187
    Microsoft .NET clients 226, 227, 231
  viewing events for 580
administrative events 571
administrator accounts 632
Administrator element 559
Administrator role 30, 433
administrators
  archiving and 598, 603
  authenticating 101
  creating Login requests for 368
  directing requests and 21
  getting job information and 61
  managing Encyclopedia and 29, 98, 124
AdminOperation arrays 181, 182, 232
AdminOperation class 181
AdminOperation element 135, 241
AdminOperation operations 181, 241
AdminOperation requests 135, 187, 231
AdminOperation type definition 241
AdminRights element 328
aggregating data 409
Aggregation data type 409
aggregation functions 409
aggregation pages 303, 307
AggregationFunctions element 409

aging rules 598, 603
See also archiving
AgreementCtl element 633
AIS. *See* Integration service
Alias element 421
alignment attributes 420
All element 495, 559
ALL logging level 531
All role 433
AllowViewTimeParameter element 443
AnalysisType element 419
analyzing data 419
Apache Ant utility 162, 532
Apache Axis code libraries 162, 163
Apache Axis environments 9, 162
Apache AXIS TCPMonitor utility 171, 177–178
Apache client sample application 170, 172
Apache Log4j utility 164
Apache log4j utility 531
Apache Logging Services Project 531
application events 581
application names 625
application programming interfaces (APIs)
archiving and 598, 604, 613
developing with xvii
implementing RSSE applications and 530, 531
logging system information and 564, 591
performance monitoring and 594
application/dime media type 16
application/soap media type 16
applications
accessing client-side bindings for 164, 219
accessing sample 530
accessing web services and 14
adding to Start menu 630
building security 530, 541, 543
building web-based 4, 77
calling remote services for 168, 221
consolidating logging information and 564, 574
customizing localized installations and 625
customizing silent installs and 628, 629, 631, 638
defining locale-specific data for 21
developing 4, 5, 170, 222
downloading files and 195, 236
enabling SOAP messaging for 164, 219
integrating with iHub 5
monitoring SOAP messages for 177
running sample 170, 222
translating messages and 4
uploading files and 109, 191, 234
writing administration 178, 226
writing batch or transaction 187, 230
archive directory 598
archive driver 598–604
archive libraries 139, 326
archive service 602
archive service command 326
archive service errors 573
archive service provider 602
archive settings 388, 599
ARCHIVE_DRIVER_JRE variable 602
archiveconfig element 600
ArchiveLibrary element 326
ArchiveLibrary property 139, 325
ArchiveOnExpiration element 410, 605
ArchiveOnExpire element 455, 605
ArchiveRoot file attribute 599
ArchiveRule data type 409, 643
ArchiveRule element 478
ArchiveRule operations
See also archiving rules
changing defaults for 608
creating folders and 125, 606
submitting jobs and 605, 606
updating rules for 607
ArchiveRule property 404
ArchiveRuleInherited element 455, 605
ArchiveRules element 297
ArchiveRules property 106, 296
ArchiveRules value 610
archives 598, 599
ArchiveServiceCmd element 326
ArchiveVolume element 602
archiving
See also archiving operations
access control lists 599
files 613
folders 125, 606, 617
notifications 598

reports 598
archiving API 613
archiving operations
See also archiving; archiving rules
configuring driver for 598, 599
copying dependent files and 599
defaults for 603
deleting files and 614
developing 598, 604
reference for 613
retrieving schedules for 139
scheduling 403, 606, 609
setting expiration policy for 603
setting root folder for 599, 600
starting 272, 610, 616
stopping 272, 614
testing for 525
archiving precedence 604
archiving rules
applying to folders 606
changing 112, 605, 608
creating 598, 603, 604
defining attributes of 409
getting 296, 610
inheriting 410, 606, 608
removing 380
scheduling jobs and 606
setting default 606
setting file-specific 380, 478
updating 381, 605, 607–609
updating files and 112, 380
uploading files and 404
archiving software 598
Argument data type 410
ArgumentList element 280
arguments 410
See also command line arguments; parameters
Arguments class 173
array definitions 412
ArrayOfColumnSchema element 498
ArrayOfCounterInfo data type 594
ArrayOfEvent data type 212
ArrayOfEventStatus data type 212
ArrayOfFileCondition data type 184, 229
ArrayOfFileInfo data type 617
ArrayOfNameValuePair class 201
ArrayOfPermission data type 617
ArrayOfResultSetSchemat element 306
ArrayOfString data type 186
ArrayOfUserAndProperties element 551
arrays 135, 359, 411, 557
ASC value 427
ascendant roles 395
ascending sort order 427
AssignedToUserId element 502
AssignedToUserName element 501
AssignedToUsersById element 396
AssignedToUsersByName element 395
AssignRolesById element 401
AssignRolesByName element 400
asterisk (*) character 16
Async value 496
asynchronous commands 272
asynchronous job IDs 503
asynchronous jobs 65, 66, 291, 337, 496, 509
asynchronous mode 361
asynchronous resource groups 66, 314, 496
asynchronous silent installations 634
AsyncResourceGroupList element 314
Attachment class 191, 234
Attachment data type 412, 643
Attachment objects 191, 235
AttachmentPart objects 192
attachments
creating 56
determining contents 412
downloading 112, 265
embedding 82, 285, 301
getting custom formats for 281
HTTP connections and 16, 108
returning search results as 78, 83
sending data as 90, 287, 316
sending files as 92, 108, 111, 265, 438
sending reports as 56, 270, 366, 517, 561
sending specific pages as 349, 355
sending to multiple locales 61
setting output formats for 367, 456
setting size 106
uploading 109, 404
AttachReportInEmail element 366, 456, 517, 561
attributes
See also properties

attributes *(continued)*
archiving and 599
binding definitions and 9
declaring web service 6
defining responses and 22
getting job 304, 308
mapping Java types and 166
Attributes element 301
Authenticate operations 546
Authenticate type definition 546
AuthenticateResponse type definition 546
authentication 42, 98, 101, 326, 530
authentication application 530, 532
authentication data 19
authentication IDs 19, 99, 172, 177, 240
authentication operations 546
authenticationSample package 530
AuthId element
Login responses 99, 172, 222, 328
SOAP headers and 19, 240, 448
SystemLogin operations 369
AuthId variable 175, 224
AuthorizationIsExternal element 525
AutoArchiveSchedule element 326
AutoArchiveSchedule property 139, 325
autoarchiving 598
See also archiving; archiving operations
autoarchiving rules. *See* archiving rules
Automatic analysis type 419
autosuggest controls 483, 484
AutoSuggest value 483
AutoSuggestThreshold element 484
AVG function 409
Axis servers. *See* Apache Axis environments

B

backslash (\) character 41, 416
backup schedules 139
BasedOnFile element 340
BasedOnFileId element 252
BasedOnFileName element 252
BasedOnObject element 315
batch applications 187, 230
batch files 162
batch operations 187, 231
batch requests 188, 231
Batch utility 162
beans 164, 165, 166
BeanSerializer objects 166
binary encoding 106
binary files 77, 78, 635, 641
binding definitions 9, 10
binding element 6, 9, 10
BIRT design files 106, 541
BIRT document files 26, 84, 87
BIRT reports 200, 349
See also reports
BIRT server. *See* iHub
BIRT viewer 201
BookMark data type 413
bookmarks 310, 413
Boolean data type 414, 429, 482
Boolean element 429, 505
Boolean parameter 505
Boolean values 414
brackets ([]) characters 416
branding 93, 625
browsers. *See* web browsers
build numbers 511, 524
build.xml 532, 541
bundling report files 268, 270, 367, 454
button controls 483

C

C/C++ applications 567, 568
C# applications 14
C# classes 219
cache 17, 155, 240, 391
cache database. *See* Caching service database
Cache-Control directive 17
Caching element 512
Caching service 334, 512, 574
Caching service database
connecting to 249, 376, 424
disconnecting from 257
getting connection information for 281
getting DBMS connection types for 282
Call objects 174, 175, 192, 196
CallOpenSecurityLibrary operations 42, 244
CallOpenSecurityLibrary requests 551
CallOpenSecurityLibrary type definition 245

CallOpenSecurityLibraryResponse type definition 245
CancelJob element 64
CancelJob operations 36, 64, 245
CancelJob type definition 245
CancelJobResponse element 65
CancelJobResponse type definition 245
CancelJobStatus data type 414
cancelled jobs 353
Cancelled value 138
CancelReport element 158
CancelReport operations 36, 157, 246
CancelReport type definition 246
CancelReportResponse element 158
CancelReportResponse type definition 246
capturing SOAP messages 177
cascading parameters 482
cascading style sheets. *See* style sheets
CascadingGroupName element 311
CascadingParentName element 482
case sensitivity 5, 346, 515, 516
CategoryPath element 420
Cell element 426
cells 425, 426
ChangesPending element 508
changing
  access control lists 545
  application names 625
  archiving rules 112, 605, 608
  file or folder privileges 114, 377, 380
  file properties 112, 377
  filter conditions 444
  folder properties 377
  images 624
  notification options 58
  passwords 516
  refresh intervals 576
  registry keys 622
  user privileges 401
Channel data type 414, 643
Channel element 249
channel icons 415
channel IDs 459
channel operations 29
ChannelCondition data type 415
ChannelField data type 416
ChannelId element 102, 276
ChannelName element 102, 276
channels
  adding subscribers to 375, 400
  adding to notification lists 389
  changing properties for 130
  creating 249
  defining attributes of 414
  deleting 256
  getting ACLs for 102, 275
  getting job information for 63
  getting privileges for 104
  getting subscribed users for 519
  handling missing 374
  naming 415
  programming tasks for 29
  removing subscribers from 375
  searching 343, 415, 416
  sending notifications to 57, 59, 60
  specifying multiple 60
  updating 130, 373, 374, 376, 415
Channels element 344
ChannelSearch data type 416
ChannelSubscriptionList property 556
character sets 16, 93
character strings. *See* strings
characters
  access control lists and 542
  multilingual reporting and 93
  not displaying 61
  passwords and 516
  search conditions and 416, 437, 445
  text string limitations for 643
  user names and 515
charset attribute 16
charts 285, 287, 300, 522
check boxes 483
child roles 395, 501
ChildRoleId element 501
ChildRoleName element 501
choice element 8
chunked attachments 16, 194
chunked messages 108
chunked transfer encoding 108, 195, 236
class files 218
class libraries 162

class paths 602
classes
building RSSE applications and 531
creating proxy objects and 14
defining search conditions and 40, 182, 227
executing reports and 199
generating C# 219
generating code libraries for 5, 216
generating from WSDL types 166
generating Java 165
implementing custom event service and 209, 210
login operations and 172, 173
mapping Java types and 166
retrieving report pages and 200
ClassId element 422, 423
ClearSystemPrinters element 403
client-initiated requests 16
CloseInfoObject operations 35, 246
CloseInfoObject type definition 246
CloseInfoObjectResponse type definition 247
Cluster element 514
cluster engine error messages 574
cluster master failover errors 573
cluster node lock violations 508
cluster nodes 151, 452, 466, 573
clusters 66, 320, 321, 514, 574
code
archive driver and 598
compiling 163
configuring RSSE logging levels and 531
creating LDAP configuration files and 533, 534
custom event web services and 209
generating 163
log consolidator application 576
logging extensions and 564, 567, 568
code emitter 162, 163
code libraries 5, 162, 163, 216
code pages 61
Collation element 491
CollationOption element 462, 492
color printers 462, 491, 492
ColorMode element 491
ColorModeOptions element 491
Column element 427
column headers 507
column headings 83, 419
column names 409, 419, 483
column schemas 421
ColumnDefinition data type 418
ColumnDetail data type 420
ColumnName element
Aggregation data type 409
as required parameter 113
DataFilterCondition data type 426
DataSortColumn data type 427
ParameterDefinition data type 483
columns
adding to queries 418
aligning data in 420
defining help text for 419
filtering values in 426
grouping 446
setting type 420
sorting on 255, 427, 512
Columns element 255
ColumnSchema data type 421
ColumnType element 113, 483
comma (,) character 542
comma-separated values files 342, 350
comma-separated values formats 341, 450, 522
command attributes 272
Command element 272
command line arguments
acencrypt utility and 632
Apache clients and 173, 188
log consolidator and 578
Microsoft .NET clients and 223
silent installs 634, 640
silent uninstalls 636, 641
command line utilities 538, 632, 635
command status attributes 272
commands (Encyclopedia) 145, 271, 272
company logos 93
company names 625
Completed folder 516, 556, 561
completed jobs 34, 551
Completed value 156
completion notices 338, 365, 517, 561
See also notifications
CompletionTime element 452, 458, 466

Component element
DataExtraction operations 255
GetContent operations 278
GetDynamicData operations 284
GetJavaReportEmbeddedComponent 301
GetJavaReportTOC operations 302
SelectJavaReportPage operations 349
SelectPage operations 355
component IDs 79, 278, 422
component names 79, 422
ComponentId element 279, 286
ComponentIdentifier data type 422
components
assigning values to 423
creating 422
determining if operational 146
getting content of 79, 278
getting embedded 82, 285, 300
logging information for 588
retrieving specific 79
searching for 82
ComponentType data type 422
composite messages 14, 41, 132
composite operations 30, 124, 132, 133
compound documents 265
Concise mode 147, 149, 334
Condition element
ChannelSearch data type 417
FileSearch data type 439
GroupSearch data type 447
JobNoticeSearch data type 460
JobScheduleSearch data type 471
JobSearch data type 473
RoleSearch data type 501
Search element and 118
UserSearch data type 518
ConditionArray element
ChannelSearch data type 417
FileSearch data type 439
GroupSearch data type 447
JobNoticeSearch data type 461
JobScheduleSearch data type 471
JobSearch data type 473
RoleSearch data type 501
UserSearch data type 519
conditions
See also search conditions
applying to jobs 137
deletion requests and 98
retrieving privileges and 104
setting filter 426
wildcards and 98
Config element 629
ConfigKey element 424
Configuration Console
enabling archive service provider 602
enabling Open Security web services 539
setting passwords for 632
setting up custom event web services and 206
setting up error logging and 566
setting up usage logging and 565
configuration error messages 574
configuration files. *See* configurations
configurations
archive drivers 598, 599
archive service provider 602
custom event web service 206
encrypted passwords 632
error logging 566, 567
external user authentication 533
external user registration 534, 538
iHub 508
install dialogs 627
localization and documentation files 636
log consolidator application 576
log consolidator database 579
Open Security applications 538
resource groups 28, 70, 71
RSSE applications 531, 532
silent installs 628, 629, 639
TCPMonitor utility 177
usage logging 565, 566
Connect requests 334
Connect value 147
connection definition files. *See* database connection definition files
connection handles. *See* ConnectionHandle element
connection objects 249, 281
connection parameters 281
connection types (DBMS) 282
ConnectionHandle element
DataExtraction operations 256

ConnectionHandle element *(continued)*
ExecuteQuery operations 268
ExecuteReport operations 47, 199, 271
FetchInfoObjectData operations 274
GetContent operations 280
GetCustomFormatData operations 281
GetDynamicData operations 285
GetEmbeddedComponent operations 287
GetJavaReportEmbeddedComponent 301
GetJavaReportTOC operations 302
GetPageCount operations 309
GetStaticData operations 317
GetStyleSheet operations 318
GetTOC operations 323
Header data type 448
ODBOTunnel operations 331
OpenInfoObject operations 332
PendingSyncJob data type 486
RunningJobs data type 503
SearchReport operations 343
SelectJavaReportPage operations 350
SelectPage operations 356
SOAP headers and 20, 156, 240
WaitForExecuteReport operations 406
ConnectionHandle variable 175, 224
ConnectionParameters element 424
ConnectionProperties element 148, 335, 360, 548
ConnectionPropertiesAreExternal element 525
ConnectionPropertyExternal element 557
connections
deleting Caching service database 257
externalizing 525
getting information about 281, 282
getting properties for 277, 548, 557
opening OLAP server 330
pinging 28, 335
preserving 20, 108, 240
setting Caching service database 249, 424
setting properties for 360
testing iHub System and 148
updating Caching service database 376
ConnectionString property 336
consolidator application. *See* log consolidator application
consolidatorconfig.xml 576
consolidatormake.xml 576
consolidatorwin.exe 578
ContainedFiles element 265, 266
content components 278
Content element
DownloadFile operations 112, 265
DownloadTransientFile operations 266
ExportParameterDefinitionsToFile operations 274
ExtractParameterDefinitionsFromFile operations 273
FileContent data type 438
SelectFiles operations 346
UploadFile operations 106, 404
content types 61
Content variable 191, 234
ContentData element 106, 413
ContentEncoding element 106, 413
ContentEncoding property 279, 287, 301
Content-ID directive 109, 111, 112
ContentId element 111, 112, 413
ContentItemList element 346
Content-Length directive 17
ContentLength element 106, 108, 413
ContentLength property 279, 287, 301
ContentRef element 279
Content-Transfer-Encoding directive 109
Content-Type directive 16, 109, 112
ContentType element 106, 112, 413, 443
context paths 207, 210, 287
Context string parameter 207
ContextPath property 201
control commands 145, 271
control type attributes 434, 483
ControlCheckBox value 483
ControlList value 483
ControlListAllowNew value 483
ControlRadioButton value 483
ControlType element 483
conversion options 283, 362, 429
ConversionOptions element 284, 367, 457
CoordinateX element 285, 287
CoordinateY element 285, 287
copy operations 132
CopyDependOnFile file attribute 599
CopyFile element 132, 244, 372
CopyFile operations 32, 131, 247

CopyFile type definition 247
CopyFromLatestVersion element 106, 107, 404
CopyFromLatestVersion variable 191, 234
copying
  archive driver configurations 601
  archives 598
  file properties 106, 107
  files 131, 247, 599
  folders 131, 247
Copyright element 630
copyright information 630
CounterId element 595
CounterIDList element 595, 596
CounterInfo data type 594
CounterInfoList element 595, 596
CounterInformation objects 594
CounterName element 595
counters 27, 28, 596
  See also performance counters
CounterValue element 595
counting
  ACL entries 293
  channel users 102
  items in folders 299
  objects 89
  records 89, 417
  report pages 87, 275, 309
  system users 102
CountLimit element
  ChannelSearch data type 417
  FileSearch data type 440
  GetChannelACL operations 276
  GetFileACL operations 293
  GetFileCreationACL operations 295
  GetParameterPicklist operations 311
  GroupSearch data type 447
  JobNoticeSearch data type 461
  JobScheduleSearch data type 472
  JobSearch data type 474
  RoleSearch data type 502
  UserSearch data type 519
CountLimit parameter 89
create operations 124
CreateActuateLogTables.sql 575, 579
CreateArchiveSubFolder file attribute 599
CreateArchiveSubFolder property 600
createCall method 196
CreateChannel element 243, 371
CreateChannel operations 249
CreateChannel type definition 249
CreateDatabaseConnection operations 35, 249
CreateDatabaseConnection type 249
CreateDatabaseConnectionResponse type definition 249
CreatedByUserId element 295
CreatedByUserName element 103, 294
CreateFileType element 243, 371
CreateFileType operations 32, 250
CreateFileType type definition 250
CreateFolder element 126, 244, 371
CreateFolder operations 32, 125, 250, 606
CreateFolder type definition 250
CreateGroup element 134, 243, 371
CreateGroup operations 34, 251
CreateGroup type definition 251
CreateNewVersion element 520
CreateNewVersion value 478
CreateParameterValuesFile operations 36, 252
CreateParameterValuesFile type definition 252
CreateParameterValuesFileResponse type definition 253
CreateQuery operations 36
CreateResourceGroup element 66, 67
CreateResourceGroup operations 66, 67, 253
CreateResourceGroup type definition 253
CreateResourceGroupResponse element 67
CreateResourceGroupResponse type definition 253
CreateRole element 126, 243, 371
CreateRole operations 42, 253
CreateRole type definition 253
CreateUser element 125, 243, 371
createUser method 179, 180
CreateUser objects 180, 188, 226, 232
CreateUser operations 42, 125, 254
CreateUser type definition 254
CreateUserRole file attribute 599
creating
  access control lists 542
  administration applications 178, 226

creating *(continued)*
archiving rules 598, 603, 604
batch applications 187, 230
channels 249
composite operations 30, 124, 132
folders 125, 250
job requests 53, 361
job schedules 423, 467, 468, 475, 525
LDAP configuration files 533, 534
localized installations 624, 625
log consolidator database 579
Login requests 99, 101, 368
notification groups 365, 444
print jobs 337
report components 422
report files 131, 477
report generation requests 46, 269
report parameters 480
resource groups 26, 65, 66, 253, 495
security roles 126, 253, 433, 499, 500
silent installs 628, 638, 639
SOAP headers 19–21
SOAP messages 9, 14, 15, 21, 164, 219
temporary files 148
transaction applications 187, 188, 231
users 125, 179, 226, 254, 543
web service interfaces 221
web-based applications 4, 77
WSDL schemas 5–11
credentials 99, 327
See also login information
Credentials element 327, 546
critical errors 567
cross-platform reporting 4, 14, 93
Crystal reports 107
CSS format 79, 521
See also style sheets
CSV element 450
CSV files 342, 350
CSV parameter 341, 522
Currency data type 428, 482
Currency element 428, 505
Currency parameter 505
CurrentTransientReportTimeout element 155, 289
custom event web service 206, 209, 210
custom events 206, 208
custom formats 88, 280
custom installation 622, 631, 638
CustomDlgs element 629, 631
CustomEvent data type 423
CustomEvent element 430, 432
CustomInputPara element 279, 355
customizing
dialog boxes 625, 627, 631
e-mail notifications 61
events 423
logging extensions 567, 568
reports 77
silent installs 629
splash screens 624, 625
CustomRef element 281

D

Daily data type 423
Daily value 469
data
aggregating 409
aligning 420
analyzing 419
defining operation-specific 96–98
deleting 98
duplicating 18
extracting 254, 283, 425
filtering 255, 420, 443
formatting 280
grouping 446
including in attachments 90, 108
localizing 5
retrieving 18, 82, 284, 286, 316
scaling 286
searching for specific 97
data components 284
data directory 635, 641
Data element 274
data extraction. *See* DataExtraction operations
data filters. *See* filters
data handlers 192
data object executable files 267, 307
data object instance files 266
data object value files 267
data rows 426, 449

data schemas 427
data sources 18
data streams 90, 286
data transfer protocols 4, 14
data type definitions 7–8, 411, 557
data type reference 212, 407, 557, 617
data types
  assigning to parameters 434, 482
  building C# classes and 219
  building JavaBeans and 165
  building RSSE applications and 557
  building web service applications and 212
  naming 8, 514
  setting filtering criteria and 443
  specifying 425, 428
  storing object arrays and 135
database connection definition files 424, 548
database connection definitions 281, 424
database drivers 575, 640
database management systems (DBMS) 282
Database property 336
database types 335
DatabaseConnection element 249, 376
DatabaseConnectionDefinition data type 424
DatabaseConnectionDefinition element 281
DatabaseEnvironment property 336
DatabaseList property 336
databases
  See also data sources
  consolidating logging information and 564, 574, 575, 576, 579
  getting connection parameters for 281
  getting connection types for 282
  installing system 633, 635, 641
  pinging 335
  programming tasks for 35
  retrieving access control lists from 541
DataCell data type 425
DataExtraction operations 35, 254
DataExtraction type definition 254
DataExtractionFormat data type 425
DataExtractionFormats element 283
DataExtractionRef element 255
DataExtractionResponse type definition 255
DataFetchHandle element 247, 274, 332
DataFilterCondition data type 426
DataLinkingURL element 285, 288
DataRef element 274
DataRow data type 426
DataRows element 449
DataSchema data type 427
DataSchema element 449
DataSortColumn data type 427
DataSource property 336
DataSourceType data type 427
DataSourceType element 484, 485
DataType data type 428
DataType element 419, 421, 434, 482
date arrays 469
Date data type 428, 482
Date element 428, 505
date formats 469
date parameters 51, 505
dated reports 598
DateOnly data type 428
DateOnly element 428, 505
DatesExcluded element 469
DaysToExpiration element 322
DBAdminPassword element 425
DBAdminUsername element 425
DBLoadPath element 425
DBMS platforms 282
DBPassword element 424
DBType property 335
DBUsername element 424
.dcd files. *See* database connection definition files
DEBUG logging level 531
DebugInstruction element 455
DecomposeCompoundDocument element 265, 266
DecomposeCompoundDocument variable 195, 236
default ACL templates 294
default archiving rules 606
Default Async value 65
default character set 16
default Encyclopedia volumes 322
default locales 93
default passwords 632
default port 11
default printer 492, 524
default resource group 65, 262
Default Sync value 65

default values 482, 631, 639
default viewer 524
DefaultEventLagTime element 431
DefaultEventPollingDuration element 431
DefaultEventPollingInterval element 431
DefaultFailureNoticeExpiration element 525, 611
DefaultObjectPrivileges property 556
DefaultOutputFileACL element 62, 305, 308
DefaultOutputFileACL property 303, 307
DefaultPrinterName element 517, 524
DefaultPrinterName property 403
DefaultSuccessNoticeExpiration element 525, 611
DefaultTableValues element 484
DefaultValue element 434, 482
DefaultValueIsNull element 482
DefaultViewingPreference element 524
DefaultViewingPreference property 403
definitions element 6
DelayFlush element 20, 240, 449
DelayFlush variable 175, 224
delete events 569
Delete operations 124, 127, 569
delete privilege 488, 620
Delete requests 98
DeleteChannel element 243, 371
DeleteChannel operations 256
DeleteChannel type definition 256
DeleteDatabaseConnection operations 35, 257
DeleteDatabaseConnection type definition 257
DeleteDatabaseConnectionResponse type definition 257
DeleteExpiredFiles operations 614
DeleteExpiredFiles type 614
DeleteExpiredFilesResponse type 614
DeleteFile element 127, 244, 372
DeleteFile operations 32, 257
DeleteFile type definition 257
DeleteFileType element 244, 371
DeleteFileType operations 32, 258
DeleteFileType type definition 259
DeleteGroup element 243, 371
DeleteGroup operations 34, 259
DeleteGroup type definition 259
DeleteJob element 244, 372
DeleteJob operations 37, 260
DeleteJob type definition 260
DeleteJobNotices element 244, 372
DeleteJobNotices operations 37, 261
DeleteJobNotices type definition 261
DeleteJobSchedule element 244
DeleteJobSchedule operations 261
DeleteJobSchedule type definition 261
DeleteResourceGroup element 73
DeleteResourceGroup operations 73, 262
DeleteResourceGroup type definition 262
DeleteResourceGroupResponse type 262
DeleteRole element 243, 371
DeleteRole operations 42, 262
DeleteRole type definition 262
DeleteUser element 128, 243, 371
DeleteUser operations 43, 263
DeleteUser type definition 263
deleting
  archiving rules 380
  Caching service connections 257
  channel subscribers 375
  channels 256
  data 98
  file dependencies 379
  file types 258
  files 127, 257, 598, 614
  folders 127, 257
  job schedules 261
  jobs 260
  list of Encyclopedia items 127
  notification groups 259
  notifications 261
  resource groups 26, 73, 262
  security roles 262, 395
  system printers 403
  user names 264
  users 128, 263, 372
delimiters 342, 350, 507
dependencies. *See* file dependencies
dependent file names 440
dependent files 410, 478, 520, 599
DependentFileId element 440
DependentFileName element 440
DependOnFiles element 619

deploying
external access control lists 540
LDAP configuration files 533, 534
reports 543–545
web services 4, 210
Depth element 323
DES value 427
descendant roles 395
descending sort order 427
Description element
Channel data type 415
ColumnDefinition data type 419
CreateFolder operations 251
File data type 436
FileInfo data type 619
Group data type 445
LicenseOption data type 474
NewFile data type 478
Printer data type 490
ResourceGroup data type 496
Role data type 499
ServerInformation data type 508
User data type 516
Description property
CopyFromLatestVersion element 106, 404
GetFolderItems operation 120
ResourceGroup element 392
ResultDef element 298
descriptions 643
design files 106
destination attributes (Ping) 333
Destination element 147, 333
Detail logging level 566, 569, 600
developing
applications. *See* applications
web-based services 4
development environments 9
development languages 14
DHTML formats 79, 521
DHTML reports 524
DHTMLPageCaching element 524
DHTMLPageCaching property 403
DHTMLPageCachingExpiration property 403
DHTMLPageCachingExpirationAge element 525
diagnostic information 147, 332
diagnostic operations 146
See also Ping operations
dialog boxes
creating silent installs and 628
customizing 625, 627, 631
encrypting information in 632
hiding 631
replacing setup images in 624
viewing default values for 631
Dimension analysis type 420
directories
Apache Axis client sample reports 163
archiving and 598, 599, 600
copying items in 131
customizing installations and 622
getting contents of 117
installing log consolidator and 575, 578
logging error information and 593
logging usage information and 564, 592
Microsoft .NET clients 216
moving items in 131
preserving workspace 113
running RSSE applications and 532
running silent installs and 635, 639
running silent uninstalls and 641
searching 116, 183, 228
directory paths. *See* paths
Disabled element 496
Disabled property 392
disk space 155
dispatch node 581
DispatchNode entry 581
display formats. *See* formats
display options
reports 367, 521
search result sets 341
displayFilterIcon property 201
DisplayFormat element 419
displayGroupIcon property 201
displaying
file properties 116
folder properties 116
Reportlets 279
reports 76, 77, 201, 520
search results 83
SOAP messages 177
specific report pages 77

displaying *(continued)*
WSDL schema definitions 11
DisplayLength element 419
DisplayName element
ColumnDefinition data type 419
ComponentIdentifier data type 422
ComponentType data type 423
FieldDefinition data type 434
ParameterDefinition data type 482
displayName element 421
DisplayType element 442
distributed environments 4
distributing reports. *See* deploying reports
DllPath property 335
DLLs 216, 564
document conversion options 283, 362, 429
document file conversions 361
document files 26, 84, 87
documentation xvii
DocumentConversionOptions data type 429
documents
See also reports
attaching to e-mail 57
creating specific versions of 54
delivering multilingual 93
downloading 265
generating 46
getting table of contents for 301
preserving workspace directories for 113
searching 82
updating 113
DoesGroupExist operations 546
DoesGroupExist type definition 547
DoesGroupExistResponse type definition 547
DoesRoleExist operations 547
DoesRoleExist type definition 547
DoesRoleExistResponse type definition 547
DoesUserExist operations 547
DoesUserExist type definition 548
DoesUserExistResponse type definition 548
.doi files. *See* data object instance files
Domain element 327
domains 99, 327
Done element 432
Done status message 49
Double data type 428, 482
Double element 428, 505
Double parameter 505
double quotation mark (") character
search results and 83, 342, 350, 507
SOAPAction directive and 17
double values 408
.dov files. *See* data object values files
DowloadEmbedded element 349
download applications 195, 236
DownloadDoubleAsBinary element 332
DownloadEmbedded element
DownloadFile operations 265
ExportParameterDefinitionsToFile operations 273
GetContent operations 279
GetCustomFormatData operations 280
GetDynamicData operations 285
GetEmbeddedComponent operations 287
GetJavaReportEmbeddedComponent 301
GetJavaReportTOC operations 302
GetStaticData operations 316
GetStyleSheet operations 317
GetTOC operations 323
SearchReport operations 342
SelectPage operations 355
DownloadEmbedded option 90
DownloadEmbedded variable 195, 236
DownloadFile class 195, 236
DownloadFile element 111
DownloadFile objects 197, 237
DownloadFile operations 32, 111, 112, 237, 264
DownloadFile requests 108, 197, 237
DownloadFile type definition 264
DownloadFileResponse attribute 91, 92
DownloadFileResponse element 111, 112
DownloadFileResponse objects 237
DownloadFileResponse type definition 265
downloading
compound documents 265
files 90, 111, 112, 194, 235, 264, 265
DownloadTransientFile operations 32, 265
DownloadTransientFile type definition 266
DownloadTransientFileResponse type definition 266
.dox files. *See* data object executable files
driver names 442, 576

DRIVER_JAR_PATH variable 602
DriverName element 442
drivers
configuring archive 598, 599
consolidating logging information and 575
creating silent installations and 640
diagnosing problems with 333
polling 455, 480
DriverTimeout element 455, 479
drop-down lists 434, 483
DroppedFromUsersById element 396
DroppedFromUsersByName element 395
DropRolesById element 401
DropRolesByName element 400
DSTAMP variable 163
Duplex element 462, 491, 492
DuplexOptions element 491
duplicate names 18, 254, 329
duplicate requests 124, 180, 249
duplicating file types 250
DurationSeconds element 452, 466
dynamic data 82, 284, 286
dynamic link libraries. *See* DLLs
DynamicDataRef element 285

E

Echo requests 149, 334
Echo value 147
elementFormDefault attribute 7
elements
binding definitions and 10
case sensitivity for 5
character limits for 643
defining data types and 8
HTTP headers and 16
mapping to Java types 166
multiple data sources and 18
SOAP messaging and 5, 15
e-mail
See also notifications
addressing 57, 125
directing to external users 552, 556, 560
overriding recipient preferences for 59, 366, 456
sending attachments with. *See* attachments
sending to multiple locales 61
setting notification options for. *See* notifications
specifying content type for 61
e-mail templates 61
EmailAddress element 516, 560
EmailAddress property 556
EmailForm property 556
EmailFormat element 367, 456
EmailWhen property 556
Embed element 286
embedded components 82, 285, 300
embedded files 438
embedded images 300, 513
embeddedDownload variable 173
EmbeddedObjPath element 522
EmbeddedProperty element 513
EmbeddedRef element 287, 301
EnableAutoParamCollection element 443
EnableColumnHeaders element 507
EnableColumnHeaders property 342, 350
EnableCustomEventService element 431
EnableFilter element 420
enableMetaData property 201
encoding 61, 404
encoding attribute 7
encoding methods 522
encoding restrictions 93
encoding schemes 106, 108
encoding style URIs 193
Encrypt attribute 632
encrypted tokens 328
EncryptedPwd element 327
encryption 632
encryption levels 368
Encyclopedia engine 147, 333
Encyclopedia Health Monitor 573
Encyclopedia service 28, 96, 147, 556
Encyclopedia volume failover errors 573
Encyclopedia volumes
adding folders to 250
assigning resource groups to 66, 496
authenticating users for 99, 326
controlling access to 98
copying objects in 131
creating archives for 598, 599, 602, 610
creating archiving rules for 603, 604

Encyclopedia volumes *(continued)*
creating items for 124–127
creating users for 125, 179, 188, 226
custom event web services and 206
defining attributes of 523
deleting items in 127–128, 598
deleting resource groups for 73
deploying reports to 543–545
downloading files from 111, 194, 235, 264, 265
executing commands for 145, 271
external registration and 530, 538
getting default 322
getting iHub options for 99
getting job information for 61
getting names 28, 321
getting properties for 139, 325
installing system database for 633, 635, 641
integrating third-party reports with 4
logging in to 99, 171, 176, 222, 326
managing 4, 29, 105, 124, 241
monitoring iHub and 151
moving items in 131
naming 524
pinging 149, 150, 335
programming tasks for 31
retrieving access control lists from 541, 543
searching 136, 182, 227
selecting items in 116, 118, 136
sending requests to 17, 21, 177, 178, 181, 241
setting system printer for 403
testing for 508
updating items in 30, 112, 128–130
updating properties for 402, 403
uploading files to 105–111, 190, 233, 404
viewing error log entries for 573
End element 494
EndArchive operations 614
EndArchive type 614
EndArchiveResponse type 615
EndTime element 494
Envelope attribute 18
environment variables 170
environments 4, 14
equal sign (=) character 542
erroneous data 18
error codes 22, 511
error event IDs 580
error events 582
error log consolidator 564, 574
error log files 568, 571, 593
error log settings 577
error logging configurations 566
Error Logging extension 564, 568, 593
error logging functions 593
ERROR logging level 531
Error Logging page 566
error messages 21, 22, 54, 56, 574
error parameters 573
error severity levels 572
error_log.csv 571
ErrorDescription element
CancelJob operations 245
CancelReport operations 246
ExecuteQuery operations 268
ExecuteReport operations 271
GetSyncJobInfo operations 319
WaitForExecuteReport operations 405
ERRORLOG_FILE_EXT property 568
ERRORLOG_FILENAME property 568
errorlogext.c 568
errors
Administrate operations and 124, 182
channel subscriptions and 374
download file operations and 237
duplicate element names and 18
failed jobs and 56
failed SOAP requests and 22
iHub status and 511
job information retrieval and 61
logging. *See* error log; error logging
Login requests and 224
namespace directives and 18
report execution and 567
RSSE applications and 531
upload file operations and 194, 235
user property update operations and 398
viewing log file entries for 572, 573
escape characters 416, 437, 445
event array 212
Event data type 212, 429

event IDs 580
event lag time setting 207
Event objects 207
event service class files 210
event service sample application 206, 209, 210
event status array 212
event status codes 209, 213, 431
event type code 584
event types 431, 584
event-based scheduling 206, 207, 208
EventList element 214
EventName element 213, 430, 467
EventNumber element 213
EventOptions data type 431
EventParameter element 213, 423, 467
events
  customizing 423
  defining attributes of 212, 429
  defining options for 431
  getting status of 213
  logging error information and 572
  logging usage information and 568, 569, 570
  monitoring RSSE applications and 531
  monitoring system 156
  scheduling reports and 206, 207, 208
  viewing administration operation 580
  viewing application 581
  viewing error 582
  viewing logging information for 583
eventSample.jar 210
EventService interface 210
EventStatus data type 213
EventStatus element 431, 467
EventStatusList element 214
EventType data type 431
EventType element 430, 467, 471
EventTypeCode entry 584
Example files 630
Examples element 630
Examples.sln 217
Excel formats 79, 81, 521
EXCEL parameter 341, 522
Excel spreadsheets 81, 280
  See also spreadsheet reports
executable file types 442
executable files
  accessing 216
  attaching to responses 270
  creating resource groups and 66
  enabling grouping for 307
  generating files from 198, 252
  getting parameters from 51, 312
  input file dependencies and 53
  running 46, 54
  setting privileges on 544
  submitting job requests for 364
  third-party reports and 4, 46, 381
  uploading 106, 107
ExecutableFileId element 268
ExecutableFileName element 486, 504
ExecutableVersionName element 487, 504
ExecutableVersionNumber element 487, 504
execute privilege 488, 620
ExecuteQuery operations 35, 266
ExecuteQuery type definition 266
ExecuteQueryResponse type definition 268
ExecuteReport application 199
ExecuteReport applications 199–200
ExecuteReport element 46, 48, 50
executeReport method 200
ExecuteReport operations
  assigning resource groups and 73
  building applications for 199–200
  defining 269
  generating reports and 37, 49, 198
  running reports and 46–48, 50
  setting response wait times for 48–49
ExecuteReport type definition 269
ExecuteReportResponse element 47, 48, 51
ExecuteReportResponse type definition 271
ExecuteReportStatus data type 432
ExecuteVolumeCommand element 146, 610
ExecuteVolumeCommand operations 31, 145, 271
ExecuteVolumeCommand type definition 271
ExecuteVolumeCommandResponse element 146, 610
ExecuteVolumeCommandResponse type definition 272
executing
  jobs 52, 54, 70, 408, 502

executing *(continued)*
queries 266
reports 46, 50, 206, 269, 405
sample applications 170, 222
silent installs 633, 634, 638
silent uninstalls 635, 636, 641
Execution element 495
execution information 570
execution requests. *See* ExecuteReport operations
execution status attributes 271, 405
ExecutionTimeout element 504
Exists element 547, 548
expiration age 606
See also archiving operations
expiration dates 606
Expiration element 415
expiration intervals 410, 516, 561, 612
expiration notices 598, 612
expiration policy 603
ExpirationAge element 410, 455, 605
ExpirationDate element 322, 455, 605
ExpirationTime element 410, 605
expired jobs 353
Expired value 138
ExpireDependentFiles element 410, 605
ExpiredFileIds element 614
ExpiredFiles element 616
ExportBeforeViewing element 442
exporting
files 442
parameter definitions 144, 273
ExportParameterDefinitionsToFile element 145
ExportParameterDefinitionsToFile operations 37, 144, 273
ExportParameterDefinitionsToFile type definition 273
ExportParameterDefinitionsToFileResponse element 145
ExportParameterDefinitionsToFileResponse type definition 274
extensible markup language. *See* XML
Extension property 250
external access control lists 540, 541
external archive software 598
external authentication 530, 533, 546
external connections 525
external data sources 548
external file types 105
external page-level security 541
external registration
authenticating users and 546
configuring LDAP files for 534
creating roles and 524
described 530
enabling 525
preparing Encyclopedia for 538
external registration application 531, 532
external security integration levels 555
external security systems 41, 530
external user names 139
external user properties 326, 550, 556
external users 547, 554, 560
ExternalProperties element 556
ExternalTranslatedRoleName data type 433
ExternalUserPropertyNames element 326
ExternalUserPropertyNames property 139, 325
ExtractParameterDefinitionsFromFile element 144
ExtractParameterDefinitionsFromFile operations 37, 143, 273
ExtractParameterDefinitionsFromFile Response element 144
ExtractParameterDefinitionsFromFile Response type definition 273
ExtractParametersFromFile type definition 273

F

–f1 command line option 634
Factory processes 65, 66, 155, 497, 504
Factory service
creating e-mail attachments and 56
creating resource groups and 65, 67, 68, 497
enabling 512
getting information about 27, 288
logging usage information for 566
overview 46
pinging 28, 147, 148, 333, 334
running jobs and 27, 152, 485, 502

FactoryPid element 504
Failed element 414, 432, 510
failed jobs 56, 432, 499
Failed message 49, 64, 158
failed requests 49
Failed value 138, 156, 511
FailNoticeExpiration property 556
failover errors 573
failure message templates 61
failure notices 339, 366, 516, 556, 561, 612
FailureNoticeExpiration element 516, 561
FATAL logging level 531
fatal logging level 572
Fault attribute 22
Fault messages 22
FeatureOptions element 100, 328
features 328
fetch direction 89
fetch handles 184, 230
FetchDirection element
  ChannelSearch data type 417
  FileSearch data type 440
  GetChannelACL operations 276
  GetFileACL operations 293
  GetFileCreationACL operations 295
  GroupSearch data type 447
  JobNoticeSearch data type 461
  JobScheduleSearch data type 472
  JobSearch data type 474
  RoleSearch data type 502
  UserSearch data type 519
FetchDirection parameter 89
FetchHandle element
  ChannelSearch data type 417
  FileSearch data type 440
  GetChannelACL operations 276, 277
  GetFileACL operations 293, 294
  GetFileCreationACL operations 103, 295
  GetFolderItems operations 299
  GroupSearch data type 447
  JobNoticeSearch data type 461
  JobScheduleSearch data type 472
  JobSearch data type 474
  RoleSearch data type 502
  SelectChannels operations 344
  SelectFiles operations 346
  SelectGroups operations 348
  SelectJobs operations 351, 352
  SelectJobSchedules operations 354
  SelectRoles operations 358
  SelectUsers operations 359
  UserSearch data type 519
FetchHandle parameter 89
FetchInfoObjectData operations 35, 274
FetchInfoObjectData type definition 274
FetchInfoObjectDataResponse type definition 274
FetchSize element
  ChannelSearch data type 417
  FileSearch data type 440
  GetChannelACL operations 276
  GetFileACL operations 293
  GetFileCreationACL operations 295
  GetParameterPicklist operations 311
  GroupSearch data type 447
  JobNoticeSearch data type 461
  JobScheduleSearch data type 472
  JobSearch data type 474
  OpenInfoObject operations 332
  RoleSearch data type 502
  SelectGroups operations 552
  SelectRoles operations 553
  SelectUsers operations 554
  UserSearch data type 519
FetchSize parameter 89
Field element
  ChannelCondition data type 416
  FileCondition data type 437
  GroupCondition data type 445
  JobCondition data type 450
  JobNoticeCondition data type 459
  JobScheduleCondition data type 468
  RoleCondition data type 500
  UserCondition data type 517
FieldControlType element 434
FieldDefinition data type 433
fields. *See* columns
FieldValue data type 435
FieldValue element 495
file access types 437
file attributes 599
File data type 435, 643
file dependencies
  archiving and 599, 604

file dependencies *(continued)*
creating files and 105, 478
executable files and 53
moving files and 329, 330
specifying 379
uploading files and 520
file descriptions 436, 442, 478
File element 265, 296, 438
file events 206
file IDs 116, 436
file lists 116
file name extensions 279, 346, 442
file names 116, 329, 346, 436, 634
File objects 192, 197, 235, 237
file operations 31
file paths. *See* paths
file permissions 114, 116, 380, 390, 599
file properties
changing 112
copying 106, 107
displaying 116
getting 116, 123, 186, 296
setting 105
updating 377
file size 436
file streams 235
file type attributes 442
file type codes 585
file type events 438
file type icons 382, 383, 442
file types
adding 250, 441, 643
archiving and 604, 606, 608
defining resource groups and 66, 68
deleting 258
duplicating 250
generating executable files and 198
getting conversion options for 283
getting parameters for 139, 142, 297
searching 346
specifying 20, 54, 240, 271, 498
updating 128, 129, 381, 382, 383
uploading files and 105
viewing logging information for 585
FileAccess data type 437, 617
FileCondition data type 437
FileCondition objects 183, 228
FileContent data type 438
file-creation privileges 103, 104
FileCreationACL template 294
FileDescription element 280
FileEvent data type 438
FileEvent element 430, 432
FileExtension element 279
FileField data type 438
FileId element
CreateDatabaseConnection operations 249
DownloadFile operations 111, 265
DownloadTransientFile operations 266
GetConnectionPropertyAssignees operations 277
GetFileACL operations 293
GetFileDetails operations 296
SaveSearch operations 340
SaveTransientReport operations 341
SetConnectionProperties operations 360
UpdateDatabaseConnection operations 376
UploadFile operations 404
FileId variable 195, 236
FileInfo data type 618
FileInfo elements 617
FileLocation element 619
FileName element
DeleteDatabaseConnection operations 257
DownloadFile operations 111, 265
GetConnectionProperties operations 548
GetConnectionPropertyAssignees operations 277
GetDatabaseConnectionDefinition operations 281
GetFileACL operations 293
GetFileDetails operations 296
Ping operations 148, 335
SetConnectionProperties operations 360
FileName variable 195, 236
FileProperties element 252, 265
FileProperties variable 195, 236
files
applying ACLs to 114
archiving 112, 604, 606, 607, 610, 613
attaching to e-mail messages 56, 92, 438
attaching to SOAP requests or responses 108, 111

bundling 268, 270, 367, 454
copying 131, 247, 599
creating 131, 477
defining attributes of 435
defining fields in 438
deleting 127, 257, 598, 614
determining if private or shared 617
downloading 90, 111, 112, 194, 235, 264, 265
embedding 91, 108, 194, 438
encrypting 632
exporting 442
exporting parameters to 144, 273
getting access rights to 103
getting ACLs for 101, 292
getting expired 615
getting parameters from 273, 312
getting privileges for 103, 104
handling missing 378
installing log consolidator and 575
localizing installations and 624
logging error information to. *See* error log files
logging RSSE objects to 531
logging usage information to. *See* usage log files
monitoring 438
moving 131, 328
naming. *See* file names
overwriting 105, 131, 330, 478
programming tasks for 31
reading 192
returning information about 344, 618
returning list of 117, 120, 344
returning location of 297
running custom installs and 623
running or printing 53
running silent installs and 628, 634, 639, 640
running silent uninstalls and 641
saving 270
searching for 116, 118, 346, 437, 439
selecting 116, 344
setting expiration policy for 598, 603
setting location of license 631
setting privileges for 114, 116, 380, 390
setting properties for. *See* file properties
specifying input 364
uninstalling localization and online documentation 636
updating 112, 113, 129, 377, 378, 381
uploading 90, 105, 109, 190, 233, 404
versioning options for 105, 519
FileSearch class 184, 229
FileSearch data type 439
FileSearch objects 183, 229
FileType data type 441, 644
FileType element
ArchiveRule data type 410, 605
CreateFileType operations 250
DocumentConversionOptions data type 429
File data type 436
FileInfo data type 619
GetDataExtractionFormats operations 283
GetDocumentConversionOptions operations 283
GetFileTypeParameterDefinitions operations 297
Header data type 449
SOAP headers and 20, 240
FileType entry 585
FileType method 183
FileType property 120, 298
FileType variable 175, 228
FileTypeCode entry 585
FileTypes element 347, 498, 510
FileTypes property 361, 392
filter conditions 426, 444
Filter element 311
FilterAdvanced value 483
FilterCriteria data type 443
filtering data 255, 420, 443
FilterList element 255
filters 117, 119
FilterSimple value 483
finding data 82, 341
See also searching
firewalls 14
FirstPage element 432
FirstPage status message 49
folder IDs 116, 298
folder lists 116
folder names 116, 251, 298

folder operations 31
FolderId element 298
FolderName element 251, 298
folders
adding privileges to 116
adding to Encyclopedia 250
applying ACLs to 114, 115
applying archiving rules to 603, 604, 606, 607, 608
archiving 125, 606, 617
changing privileges for 114
changing properties 377
copying 131, 247
creating 125, 250
customizing silent installs and 630
deleting 127, 257
getting ACLs for 101, 292
getting archiving rules for 610
getting files in 120, 121, 297, 344
getting properties for 116, 123
handling missing 378
listing available 117
moving 131, 328
programming tasks for 31
running online archive driver and 598, 599, 600
searching 120, 122
setting expiration policy for 603
setting home 125, 516, 556
updating 112, 377, 378, 381
FolderWhen property 556
foreign keys 580
Format element 287, 332, 521
format type attributes 299, 444
FormatList element 299
formats
converting report instances and 361
creating job schedules and 469
displaying dynamic data and 287
displaying reports and 79, 419, 521
displaying search results and 83, 341
e-mail attachments and 57, 59, 367, 456
extracting data and 283, 425
generating locale-specific data and 5
getting conversion options for 284
getting custom 88, 280
getting display 87
getting supported 27, 299
localizing reports and 21, 93
MAPI encoding and 61
overriding preferences for 56
retrieving information objects and 449
running jobs and 56
searching reports and 522
specifying component IDs and 278
specifying text 420
specifying type 299, 444
FormatType data type 444
FormatType element 299
FormName element 463
freeing resources 567
FrequencyInDays elements 423
FrequencyInMonths element 476
FrequencyInWeeks element 526
functions
aggregating data and 409
retrieving error information and 593
retrieving usage information and 592
searching external security systems and 41
fundamental data types. *See* data types

G

GeneralDlgs element 629, 631
generating
C# classes 219
code libraries 5, 216
Java classes 165
JavaBeans 165, 166
Javadoc 163
locale-specific data 5
open server reports 35
report object value files 252, 270
reports 46, 54, 198
source code 163
table of contents 85
third-party reports 46
XML documents 80
Generation element 512
generation events (reports) 570
generation requests
assigning to resource groups 21, 240
cancelling 49, 64, 156, 157, 405, 414
creating 46, 269

prioritizing 556
retrying 498
scheduling 54
setting status of 432
setting wait intervals for 48, 49, 268, 270
submitting jobs for 365
Get operations 136
Get requests 138
getActuateSoapPort method 169
getActuateSoapPortAddress method 169
GetAllCounterValues operations 595
GetAllCounterValues type definition 595
GetAllCounterValuesResponse type definition 595
GetAllPaperSizes element 320
getAuthId method 177
GetChannelACL element 102, 104
GetChannelACL operations 43, 102, 104, 275
GetChannelACL type definition 275
GetChannelACLResponse element 103, 104
GetChannelACLResponse type definition 276
GetConnectionProperties operations 548, 557
GetConnectionProperties type definition 548
GetConnectionPropertiesResponse type definition 548
GetConnectionPropertyAssignees operations 43, 277
GetConnectionPropertyAssignees type definition 277
GetConnectionPropertyAssigneesResponse type definition 277
GetContent element 80
GetContent operations 37, 76, 79, 278
GetContent type definition 278
getContentData method 202
GetContentResponse element 81
GetContentResponse type definition 279
GetCounterValues operations 595
GetCounterValues type definition 595
GetCounterValuesResponse type definition 596
GetCustomFormat element 88, 280
GetCustomFormat method 37, 88, 280
GetCustomFormat operations 37, 76, 88, 280
GetCustomFormatResponse type definition 281
GetDatabaseConnectionDefinition operations 35, 281
GetDatabaseConnectionDefinition type definition 281
GetDatabaseConnectionDefinitionResponse type definition 281
GetDatabaseConnectionParameters operations 35, 281
GetDatabaseConnectionParameters type definition 282
GetDatabaseConnectionParametersResponse type definition 282
GetDatabaseConnectionTypes operations 35, 282
GetDatabaseConnectionTypes type definition 282
GetDatabaseConnectionTypesResponse type definition 282
GetDataExtractionFormats operations 32, 283
GetDataExtractionFormats type definition 283
GetDataExtractionFormatsResponse type definition 283
GetDocumentConversionOptions operations 37, 283
GetDocumentConversionOptions type definition 283
GetDocumentConversionOptionsResponse type definition 284
GetDynamicData operations 37, 284
GetDynamicData type definition 284
GetDynamicData value 286
GetDynamicDataResponse type definition 285
getEmbed method 202
GetEmbeddedComponent element 82
GetEmbeddedComponent operations 37, 76, 82, 285
GetEmbeddedComponent type definition 285
GetEmbeddedComponentResponse element 82
GetEmbeddedComponentResponse type definition 287
GetEventStatus data type 213
GetEventStatus method 209, 210

GetFactoryServiceInfo element 155, 288
GetFactoryServiceInfo operations 288
GetFactoryServiceInfo type definition 288
GetFactoryServiceInfoResponse element 156
GetFactoryServiceInfoResponse type definition 288
GetFactoryServiceJobs element 153
GetFactoryServiceJobs operations 289
GetFactoryServiceJobs type definition 290
GetFactoryServiceJobsResponse element 154
GetFactoryServiceJobsResponse type definition 292
GetFileACL element 102
GetFileACL operations 43, 101, 292
GetFileACL type definition 292
GetFileACLResponse element 102
GetFileACLResponse type definition 293
GetFileCreationACL element 103, 104
GetFileCreationACL operations 43, 103, 104, 294
GetFileCreationACL type definition 294
GetFileCreationACLResponse element 103, 104
GetFileCreationACLResponse type definition 295
GetFileDetails element 123, 610
GetFileDetails operations
  archiving and 610
  defining 296
  generating reports and 32, 123
  returning properties and 123
GetFileDetails type definition 296
GetFileDetailsResponse element 123, 611
GetFileDetailsResponse type definition 296
GetFileTypeParameterDefinitions element 139, 143
GetFileTypeParameterDefinitions operations 33, 142, 297
GetFileTypeParameterDefinitions type definition 297
GetFileTypeParameterDefinitionsResponse element 143
GetFileTypeParameterDefinitionsResponse type definition 297
GetFolderItems element 121, 122
GetFolderItems operations
  defining 297
  managing report files and 33
  returning properties and 120, 121
  searching and 116, 122, 183, 228
GetFolderItems type definition 297
GetFolderItemsResponse element 121, 122
GetFolderItemsResponse type definition 298
GetFormats element 87
GetFormats operations 76, 87, 299
GetFormats type definition 299
GetFormatsResponse element 88
GetFormatsResponse type definition 299
GetInfoObject operations 35, 299
GetInfoObject type definition 300
GetInfoObjectResponse type definition 300
GetJavaReportEmbeddedComponent operations 38, 300
GetJavaReportEmbeddedComponent Response type definition 301
GetJavaReportEmbeddedComponent type definition 300
GetJavaReportTOC operations 38, 301
GetJavaReportTOC type definition 301
GetJavaReportTOCResponse type definition 302
GetJobDetails element 61, 75
GetJobDetails operations 38, 61, 75, 302
GetJobDetails type definition 302
GetJobDetailsResponse element 62, 75
GetJobDetailsResponse type definition 304
getMessageContext method 197
GetMetaData operations 35, 305
GetMetaData type definition 305
GetMetaDataResponse type definition 306
GetNextExpiredFiles operations 615
GetNextExpiredFiles type 615
GetNextExpiredFilesResponse type 616
GetNoticeJobDetails element 63
GetNoticeJobDetails operations 38, 63, 306
GetNoticeJobDetails type definition 306
GetNoticeJobDetailsResponse type definition 307
GetPageCount element 87
GetPageCount operations 38, 76, 87, 275, 309
GetPageCount type definition 309
GetPageCountResponse element 87
GetPageCountResponse type definition 309
GetPageNumber operations 38, 310

GetPageNumber type definition 310
GetPageNumberResponse type definition 310
GetParameterPickList operations 38, 310
GetParameterPickList type definition 310
GetParameterPickListResponse type definition 311
GetReportParameters element 51, 54
GetReportParameters operations 38, 51, 312
GetReportParameters type definition 312
GetReportParametersResponse element 52
GetReportParametersResponse type definition 312
GetResourceGroupInfo element 70
GetResourceGroupInfo operations 70, 313
GetResourceGroupInfo type definition 313
GetResourceGroupInfoResponse element 70
GetResourceGroupInfoResponse type definition 313
GetResourceGroupList element 69
GetResourceGroupList operations 69, 314
GetResourceGroupList type definition 314
GetResourceGroupListResponse element 69
GetResourceGroupListResponse type definition 314
getResponseMessage method 197
GetSavedSearch operations 41, 315
GetSavedSearch type definition 315
GetSavedSearchResponse type definition 315
getSerializer method 166
GetServerResourceGroupConfiguration element 71
GetServerResourceGroupConfiguration operations 70, 315
GetServerResourceGroupConfiguration Response element 71
GetServerResourceGroupConfiguration Response type definition 316
GetServerResourceGroupConfiguration type definition 315
GetStaticData operations 38, 316
GetStaticData type definition 316
GetStaticData value 286
GetStaticDataResponse type definition 317
GetStyleSheet operations 38, 76, 317
GetStyleSheet type definition 317
GetStyleSheet value 286
GetStyleSheetResponse type definition 318
GetSyncJobInfo element 156
GetSyncJobInfo operations 38, 318
GetSyncJobInfo type definition 318
GetSyncJobInfoResponse element 157
GetSyncJobInfoResponse type definition 318
GetSystemMDSInfo operations 319
GetSystemMDSInfo type definition 319
GetSystemMDSInfoResponse type definition 319
GetSystemPrinters element 139, 140
GetSystemPrinters operations 320
GetSystemPrinters type definition 320
GetSystemPrintersResponse element 140
GetSystemPrintersResponse type definition 320
GetSystemServerList element 151
GetSystemServerList operations 320
GetSystemServerList type definition 321
GetSystemServerListResponse element 152
GetSystemServerListResponse type definition 321
GetSystemVolumeNames operations 321
GetSystemVolumeNames type definition 321
GetSystemVolumeNamesResponse type definition 321
GetTOC element 86
GetTOC operations 39, 76, 85, 322
GetTOC type definition 322
GetTOCResponse element 86
GetTOCResponse type definition 323
GetTranslatedRoleNames operations 549
GetTranslatedRoleNames type definition 549
GetTranslatedRoleNamesResponse type definition 549
GetTranslatedUserNames type definition 549
GetTranslatedUserNamesResponse type definition 549
GetUserACL operations 550
GetUserACL type definition 550
GetUserACLResponse type definition 550
GetUserLicenseOptions operations 43, 323
GetUserLicenseOptions type definition 323
GetUserLicenseOptionsResponse type definition 324
GetUserPrinterOptions element 139, 142
GetUserPrinterOptions operations 39, 324

GetUserPrinterOptions type definition 324
GetUserPrinterOptionsResponse element 142
GetUserPrinterOptionsResponse type definition 325
GetUserProperties operations 550
GetUserProperties type definition 550
GetUserPropertiesResponse type definition 551
GetUsersToNotify element 551
GetUsersToNotify operations 551
GetUsersToNotifyResponse type definition 551
GetVolumeProperties element 138, 139
GetVolumeProperties operations 31, 139, 325
GetVolumeProperties type definition 325
GetVolumePropertiesResponse element 140
GetVolumePropertiesResponse type definition 325
grant privilege 488, 620
GrantedRoleId element
  GetChannelACL operations 104, 276
  GetFileACL operations 293
  GetFileCreationACL operations 295
  PrivilegeFilter data type 493
GrantedRoleName element
  GetChannelACL operations 104, 276
  GetFileACL operations 293
  GetFileCreationACL operations 295
  PrivilegeFilter data type 493
GrantedUserId element
  GetChannelACL operations 104, 276
  GetFileACL operations 293
  GetFileCreationACL operations 295
  PrivilegeFilter data type 493
GrantedUserName element
  GetChannelACL operations 104, 276
  GetFileACL operations 293
  GetFileCreationACL operations 295
  PrivilegeFilter data type 493
GrantPermissions element 115, 375, 380
GrantPermissions operations 115
graphics 93, 513, 624
graphs. *See* charts
Group data type 444, 644
Group element 251, 482, 485
group operations 34
Group property 50
GroupCondition data type 445
GroupField data type 445
GroupHeadingFields element 446
Grouping data type 446
grouping pages 303, 307
GroupingEnabled element
  GetJobDetails operations 303
  GetNoticeJobDetails operations 307
GroupKey element 446
GroupName element 547, 554
groups
  See also notification groups; resource groups
  creating user 251
  deleting 259
  programming tasks for 34
  searching 347, 445, 446
  testing for external 546
Groups element 348, 553
GroupSearch data type 446
GroupSortOrder element 446

H

HasMore element 616
Header class 224
Header data type 448
header elements (messages) 9
  See also SOAP headers
Heading element 419
Headline element 364, 458
headlines 364, 458, 466
health monitoring errors 573
helper classes 531
HelpText element 419, 482
hexadecimal values 408
hidden objects 440
hidden parameters 54, 434, 482
home folder 125, 516, 556
HomeFolder element 516, 560
HomeFolder property 556
HorizontalAlignment element 420
Host directive 16
Host property 336
HostString property 336
HTML reports 20

HTTP connections
accessing iHub and 26
chunked transfer encoding and 108
determining version for 16
embedding files and 108
sending and receiving over 14, 90
HTTP headers 15, 16, 109, 112
hypercharts 522
hyperlinks
See also URLs
getting context paths for 287
redirecting 287
retrieving dynamic data and 285
sending output files and 57, 366
setting chart 285, 522
hypertext transfer protocol. *See* HTTP

I

icons
defining channel 415
setting file type 383
specifying URLs for 442
Id element
as search criteria 116
Channel data type 415
ComponentIdentifier data type 422
ComponentType data type 422
CopyFile operations 248
DatabaseConnectionDefinition type 424
DeleteChannel operations 256
DeleteFile operations 258
DeleteGroup operations 260
DeleteJob operations 260, 262
DeleteRole operations 263
DeleteUser operations 264, 373
File data type 436
FileInfo data type 618
Group data type 445
MoveFile operations 330
ObjectIdentifier data type 479
Role data type 499
SelectChannels operations 344
SelectFiles operations 345
SelectGroups operations 348
SelectJobs operations 351
SelectJobSchedules operations 353
SelectRoles operations 357
SelectUsers operations 359
UpdateChannel operations 374
UpdateFile operations 378
UpdateGroup operations 384
UpdateJobSchedule operations 386
UpdateRole operations 393
UpdateUser operations 398
User data type 515
Id parameter 96, 182, 227
ID property 120
IDAPI. *See* Information Delivery API
IDAPI applications 170, 222
See also applications
idapi.jar 163
IDE 5
identifiers 17
IdList element
as search criteria 116
CopyFile operations 248
DeleteChannel operations 256
DeleteFile operations 127, 258
DeleteGroup operations 259
DeleteJob operations 260, 262
DeleteRole operations 263
DeleteUser operations 264, 373
MoveFile operations 329
SelectChannels operations 343
SelectFiles operations 345
SelectGroups operations 348
SelectJobs operations 351
SelectJobSchedules operations 353
SelectRoles operations 357
SelectUsers operations 359
UpdateChannel operations 374
UpdateFile operations 378
UpdateGroup operations 384
UpdateJobSchedule operations 386
UpdateRole operations 393
UpdateUser operations 397
IdList parameter 96, 182, 227
IDS_MSG_COPYING value 627
IDS_SETUP_FINISH_MSG value 627
IgnoreActiveJob element 261, 262
ignoreDup argument 188, 231
IgnoreDup element
CreateChannel operations 249

IgnoreDup element *(continued)*
CreateFileType operations 250
CreateFolder operations 251
CreateGroup operations 252
CreateRole operations 254
CreateUser operations 254
error conditions and 124
UpdateChannel operations 374
UpdateFileType operations 382
UpdateGroup operations 384
UpdateRole operations 394
UpdateUser operations 398
IgnoreMissing element
DeleteChannel operations 256
DeleteFile operations 258
DeleteFileType operations 259
DeleteGroup operations 260
DeleteJob operations 261, 262
DeleteRole operations 263
DeleteUser operations 264, 373
error conditions and 124
UpdateChannel operations 374
UpdateFile operations 378
UpdateFileType operations 382
UpdateGroup operations 384
UpdateJobSchedule operations 387
UpdateRole operations 394
UpdateUser operations 398
iHub
See also servers
accessing 26
assigning resource groups to 66, 71, 497, 509
authenticating users for 101, 326
defining attributes of 507
defining version information for 511
deleting expired files on 614
deleting resource groups for 73
getting available options for 99
getting capacity of 155
getting list of 27, 151
getting resource groups for 27, 69, 70, 314
getting state of 320
getting supported formats for 27, 299
getting supported locales for 27
getting system printer for 27, 320
getting version of 322
installing log consolidator for 574, 576
logging error information for 566, 568, 572, 593
logging in to 368
logging resource usage for 564
logging usage information for 565, 567, 570, 592
naming 508, 509
pinging 332
programming tasks for 26
running custom installs for 638
running jobs and 65, 206, 207
running RSSE applications on 532, 540
running silent installs for 628, 639, 640
running silent uninstalls for 641
sending SOAP messages to 11, 14, 21, 240
setting response times for 48
setting state 510
setting status 510
updating resource groups for 361, 392
viewing error log entries for 573
viewing error messages for 574
iHub Integration Technology
implementing logging extensions and 564, 567, 568
implementing RSSE interface and 530
running custom installs for 638
running silent installs for 628, 639, 640
running silent uninstalls for 641
iHub objects 178, 226
iHub operations 26
iHub service type codes 584, 587
iHub services 152, 512, 573
See also specific service
iHub System
authenticating users for 99, 101
creating silent installs for 628, 638, 639
customizing installations for 622, 638
defining type 513
encoding restrictions for 93
getting licensing options for 323, 519
localizing installations for 624, 625
monitoring 151–156
setting licensing options for 474
tracking usage and error information for 588
uninstalling 635, 641

image components 82, 300
image IDs 202
ImageMapURL format 80, 521
ImageMapURL value 287
images 93, 202, 513, 624
immediate jobs 365
InActive element 414
InActive message 64, 158
IncludeFolder element 617
IncludeHiddenObject element 440
IncludeInheritedPrivilege element 417
INFO logging level 531
InfoObject element 300, 428
InfoObject value 484
InfoObjectData data type 449
InfoObjectDataFormat data type 449
InfoObjectId element 300
InfoObjectName element 300
Information Console 77, 628, 643
Information Delivery API
  Apache Axis clients and 162, 164
  archiving and 598, 604–612, 613
  constructing composite messages and 14, 41, 132
  data type reference for 212, 407, 617
  developing with 4, 5, 170, 222
  HTTP transmissions and 26
  installing archive driver and 598
  login mechanisms for 98
  Microsoft .NET clients and 216, 219
  multilingual reporting and 93
  operations reference for 239
  required libraries for 162
  retrieving multiple objects and 89
  searching external users and 41
  SOAP messaging protocol for 14, 15, 90
  text string maximum lengths 643
information object file types 427
information objects
  closing 246
  defining data formats for 449
  getting 299
  opening 331
  programming tasks for 35
  querying 300
  retrieving data from 274, 449
  searching 343
  storing parameters in 427, 484, 485
  submitting job requests for 361
Information service 574
informational messages 572
InheritedFrom element 410, 605, 607
inheriting archive rules 606, 608
–input command line option 632
Input element 551
input element 10
input file IDs 54
input files
  encrypting 632
  executing reports and 198
  setting version numbers for 54
  specifying 364
  submitting jobs and 53, 54, 452, 465
input message element 9
input messages 9, 10, 167, 221
  See also requests
input parameters 279, 388, 639
InputDetail element 61, 304, 308
InputDetail property 303, 306
InputFile element 270
InputFileId element
  ExecuteQuery operations 267
  ExecuteReport operations 270
  JobProperties data type 465
  JobScheduleSearch data type 471
  JobSearch data type 473
  PrintReport operations 338
  SubmitJob operations 364
InputFileName element
  ExecuteQuery operations 267
  ExecuteReport operations 269
  JobProperties data type 466
  JobScheduleSearch data type 471
  JobSearch data type 473
  PrintReport operations 338
  SubmitJob operations 54, 364
InputFileVersionName element 466
InputParameter element 245
installation
  Apache Ant utility 532
  archive driver 598, 601
  customizing 622, 638
  Localization and Online Documentation 636

installation *(continued)*
localizing 624, 625
log consolidator application 574, 575, 576
logging extensions 564
Page Security application 540, 541
RSSE applications 531, 532
testing 624
installation libraries 624
installation scripts (UNIX) 638, 639
InstallShield utility 624, 625, 627, 628
Integer data type 428, 449, 482
Integer element 428, 505
Integer parameter 505
integers 449
integrated development environment 5
See also development environments
Integration element 512
Integration service 334, 512, 571
IntegrationLevel element 555
IntervalInSeconds element 494
invalid locales 93
invalid user names 124
invisible characters 61
IP addresses 207, 475
IsAdHoc element 52, 113, 483
IsAutoArchiveRunning element 525
IsBundled element 268, 270, 367, 454
IsCab utility 627, 628
IsColor element 462, 492
IsCompoundDoc element 443
IsDefaultPrinter element 492
IsDynamicSelection element 484
iServer. *See* iHub
IsExecutable element 442
IsExecutable property 250
IsHidden element 434, 482
IsInherited element 410, 605, 607
IsLoginDisabled element 516
IsNative element 442
IsNative property 250
IsPassword element 482
IsPrintable element 442
IsPrintable property 250
IsProgressive element 503
IsReportCompleted element 309
IsRequired element 434, 442, 482
IsSyncFactory element 504
IsSyncJob element 503
IsTransient element 486, 503
IsViewParameter element 484
ItemList element 298, 346
iterators 197

J

Jakarta Commons code libraries 164
JAR files 14, 163, 532, 575
Java Architecture for XML Binding 575
Java classes 165
Java objects 575
Java RSSE framework 530
See also RSSE applications
Java Runtime Environment 602
Java stubs 168
Java types 166
JavaBeans 164, 165, 166
JavaBeans Activation Framework 164
javac compiler 163
Javadoc 163
JavaMail code libraries 164
JavaServer Pages 77
JAXB framework 575
JDBC drivers 575, 576
job attributes 408
job details 452
job events 206, 450
job IDs 56, 64, 465
job names 465, 469
job operations 36
See also jobs
job printer settings 338, 461
job purging errors 573
job state attributes 138, 452, 458, 465, 470
job status attributes 271, 319
job type attributes 451, 465, 470, 496
job type code 581, 585
job type descriptions 585
JobAttributes element 61, 63, 304, 308
JobAttributes property 303
JobCondition data type 450
JobEvent data type 450
JobEvent element 430, 432
JobField data type 451

JobId element
CancelJob operations 245
GetJobDetails operations 303
GetNoticeJobDetails operations 306
GetReportParameters operations 312
JobEvent data type 451
JobNotice data type 458
JobProperties data type 465
PrintReport operations 339
RunningJobs data type 503
SubmitJob operations 368
JobInputDetail data type 452
JobName element
ExecuteQuery operations 267
ExecuteReport operations 269
JobEvent data type 451
JobNotice data type 458
JobProperties data type 465, 469
PrintReport operations 338
SubmitJob operations 364
JobNotice data type 457, 644
JobNoticeCondition data type 459
JobNoticeField data type 460
JobNotices element 352
JobNoticeSearch data type 460
JobPrinterOptions data type 461, 644
JobProperties data type 463, 644
JobRetryInterval element 524
JobRetryInterval property 403
jobs
archiving 608
assigning resource groups to 65, 74, 465
cancelling 64, 245, 414
changing autoarchiving rules and 605
changing priorities for 129
defining pending 485
deleting 260
event-based scheduling and 206
expiring notifications for 611, 612
failing 56
getting iHub capacity for 155
getting information about 137, 152, 156, 289, 318
getting parameters for 51, 305, 312
getting properties of 61, 63, 137, 302, 306
getting resource groups for 75
monitoring performance of 450
preserving status information for 367
prioritizing 66, 129, 338, 364
programming tasks for 36
removing resource groups and 262
removing schedules 261
restarting iHub and 54
retrying 403, 454, 494, 498, 499
running 52, 54, 70, 408, 502
scheduling 423, 467, 468, 475, 525
searching for 450, 467, 469, 470, 472
selecting 350, 353
sending notifications for 56, 57, 58, 351, 457
setting printer options for 338, 461
setting properties for 463
setting state 138, 452, 458, 465, 470
setting status of 271, 319, 432
submitting 53, 361
updating schedules for 386, 387, 391
viewing logging information for 585
Jobs element 351, 354
JobSchedule data type 467, 645
JobScheduleCondition data type 467
JobScheduleDetail data type 468
JobScheduleField data type 469
JobScheduleSearch data type 470
JobSearch data type 472
JobState element 458
JobStatus element 451
JobType element 451, 465, 470
JobTypeCode entry 581, 585
JobTypeDescription entry 585
JRE (Java Runtime Environment) 602
JSP pages 77

K

KeepOutputFile element 367, 457
KeepWorkingSpace element 479
KeepWorkspace element 455

L

Label element 421
Lag time parameter 207
LagTime element 213, 430
language-independent reporting 93
large files 108

large icons 383, 415, 442
LargeImageURL element 415, 442
LatestVersionOnly element
  CopyFile operations 132, 248
  DeleteFile operations 258
  GetFolderItems operations 298
  MoveFile operations 330
  Search element and 131
  SelectFiles operations 345
  UpdateFile operations 377
LDAP authentication application 530, 532
LDAP configuration files 533, 534
LDAP drivers 640
LDAP external registration application 531, 532
LDAP servers 41, 530, 533, 538
ldapSample package 531
libraries
  accessing third-party code 163
  accessing web services and 14
  Apache Axis environments and 162
  calling RSSE API 244
  compiling schemas 162
  generating code 5, 216
  getting archive 139, 326
  localizing installations and 624, 625, 627
  migrating RSSE applications and 540
  running logging extensions and 564
library files 163
license files 631
license keys 322
license options 323, 474, 519
LicenseFileCtl element 631
LicenseFileDlg element 631
LicenseOption data type 474
LicenseOptions element 326, 560
Lightweight Directory Access Protocol. *See* LDAP servers
links 56
  See also hyperlinks; URLs
Linux servers
  configuring online archive provider for 602
  customizing installations for 638
  running silent installs for 638, 639, 640–641
  running silent uninstalls for 641
list controls 434, 483
List element 282
listening port 177, 217
literal attribute 10
literal messages 10
Locale element
  Attachment data type and 413
  Header data type 448
  SelectPageResponse element and 78
  SOAP headers and 21, 240
  UploadFile requests and 106
locale information 19
locale keys 625
Locale parameter 5, 93
Locale property 279, 287, 301
Locale variable 175, 224
locales
  customizing installations for 624, 625
  defined 93
  developing for 4
  formatting data for 21
  generating reports for 5, 93
  getting supported 27, 299
  sending attachments and 413
  sending e-mail to multiple 61
  setting invalid 93
  specifying 21
  updating job schedules for 388
  viewing specific pages and 78
LocalExtension element 442
localhost parameter 10, 217
Localization and Online Documentation files 628, 636
locating data 82, 341
  See also searching
Location element 490
location settings 388
locks 508
log consolidator application 564, 574–589
log consolidator command line arguments 578
log consolidator database 574, 579
log consolidator database tables 580–589
log consolidator schema 580, 589
log files
  archiving report files and 600
  naming 566, 567, 568

running silent installs and 628, 634, 635, 640
running silent uninstalls and 636, 641
setting paths for 592, 593
tracking error and usage information and 564, 567, 568
log.properties file 531
log4j API 531
LogConsolidator element 577
LogFile element 555
logging
error information 564, 566, 574, 593
RSSE objects 531
usage information 564, 565, 574, 592
logging APIs 564
Logging Consolidator service 579
logging counters 596
logging extensions 564, 567, 568, 593
logging functions 592, 593
logging in to
Configuration Console 632
Encyclopedia volumes 99, 171, 176, 222, 326
iHub 368
logging levels
error logs 566, 571
online archive driver 600
RSSE applications 531
usage logs 566, 569
logging refresh intervals 576
logging tool 531
logical values 414
login actions 171, 222
login applications 172, 173, 176
Login class 166, 220
Login element 100
login information 548
See also credentials
login method 168, 176, 221
Login objects 176, 224
login operations
See also SystemLogin operations
authenticating users and 43, 99
defining 171, 222, 326
generating authentication IDs for 19, 172, 222
Login pages 321
Login requests
creating 99
disabling 516
embedding authentication IDs in 177
input and output child elements in 10
sending 172, 224
type definitions for 8
Login responses 99, 172, 222, 225
login settings 99
Login type definition 8, 165, 220, 326
LoginResponse element 100
LoginResponse objects 176
LoginResponse type definition 327
logins 167
LogLevel file attribute 600
$LOGNAME variable 640
logos 93
LongDescription element 442

M

mail. *See* e-mail
mail transfer protocols 4
MaintenanceTypeCtl element 636
Management Console 524, 643
Manufacturer element 489
MAPI notifications 4, 61
MasterPage property 201
Match element
ChannelCondition data type 416
FileCondition data type 437
GroupCondition data type 445
JobCondition data type 450
JobNoticeCondition data type 459
JobScheduleCondition data type 468
RoleCondition data type 500
UserCondition data type 517
MAX function 409
MaxFactory element 497, 509
MaxFactory property 361, 392
MaxFactoryProcesses element 289
MaxFiles element 616
MaxHeight element 279, 355
MaxJobPriority element 516, 561
MaxJobRetryCount element 524
MaxJobRetryCount property 403
MaxPriority element 497

MaxPriority property 392, 556
MaxRetryCount element 454, 499
MaxSyncJobRuntime element 155, 289
MaxVersions element
  CopyFile operations 248
  JobInputDetail data type 455
  MoveFile operations 131, 330
  NewFile data type 478
  UploadFile operations 105
MDSInfo data type 474
MDSInfoList element 319
MDSIPAddress element 475
MDSPortNumber element 475
Measure analysis type 420
media types 16, 61
MemberOfGroupId element 519
MemberOfGroupName element 519
Message Distribution service
  defining attributes of 474
  enabling 512
  getting information about 27, 319
  pinging 28, 147, 333
message element 6, 9
message IDs 449
message namespace declaration 7
MessageContext objects (downloads) 197
messages
  See also error messages; status messages
  accessing multiple data sources and 18
  Administrate operations and 124
  binding to operations 5, 9, 10, 167, 221
  cancelling report generation requests and 158
  capturing 177
  chunking 108
  combining transactions in 30
  creating locale-specific 93
  creating silent installs and 628
  creating SOAP 9, 14, 15, 21, 164, 219
  cross-platform applications and 4
  decoding and encoding 166
  defining composite 14, 132
  defining literal 10
  displaying SOAP 177
  embedding attachments in 265
  embedding data in 90
  embedding files in 91, 108
  encoding scheme for 7
  grouping operations in 132
  logging in to Encyclopedia and 167, 172, 174, 221, 222
  parsing 19
  referencing namespace declarations in 18
  required elements of 17, 19
  search operations and 41
  sending and receiving 14, 90
  sequence of elements in 8
  setting length 17
  specifying media types for 16
  streaming reports with 190, 233
  uploading files and 109
Messaging Application Programming Interface. *See* MAPI notifications
messaging systems 4, 14
metadata 139, 166, 305
Microsoft .NET clients 216–219
Microsoft .NET environments 9
Microsoft Active Directory servers 41, 530
Microsoft C# development environments 216
Microsoft Exchange messaging protocol 4
Microsoft Windows. *See* Windows systems
MIME attachments 190, 194, 233
MIME boundaries 109, 112
MIME encoding 16
MIME headers 109
MIME_boundary directive 78, 80, 109
MimeType element 425, 429
MimeType property 279, 287, 301
MIN function 409
MinFactory element 497, 509
MinPriority element 496
MinPriority property 392
missing channels 374
missing files or folders 378
mode attributes (Ping) 334
Mode element 147, 334
Model element 489
MonitoredFilePath element 438
monitoring counters 596
  See also performance counters
monitoring SOAP messages 177
monitoring tools 564, 594
  See also performance monitoring
Monthly data type 475

Monthly value 469
MoveFile element 131, 244, 372
MoveFile operations 33, 131, 328
MoveFile type definition 328
multilingual reporting 93
See also locales
multipart/related media type 16
multithread tests 567, 568, 592, 593
MutexClass element 442

N

Name element
Argument data type 410
as search criteria 116
Channel data type 415
ColumnDefinition data type 419
ColumnSchema data type 421
ComponentIdentifier data type 422
ComponentType data type 423
CopyFile operations 248
DatabaseConnectionDefinition type 424
DeleteChannel operations 256
DeleteFile operations 258
DeleteFileType operations 259
DeleteGroup operations 260
DeleteResourceGroup operations 262
DeleteRole operations 263
DeleteUser operations 264
FieldDefinition data type 434
FieldValue data type 435
File data type 436
FileInfo data type 618
FileType data type 442
FilterCriteria data type 443
GetResourceGroupInfo operations 313
Group data type 445
LicenseOption data type 474
MoveFile operations 330
NameValuePair data type 477
NewFile data type 477
ObjectIdentifier data type 479
ParameterDefinition data type 482
ParameterValue data type 485
Printer data type 489
PropertyValue data type 493, 559
ResourceGroup data type 496
Role data type 499
SelectChannels operations 343
SelectFiles operations 345
SelectFileTypes operations 347
SelectGroups operations 348
SelectRoles operations 357
SelectUsers operations 359
SortColumn data type 513
Stream data type 513
UpdateChannel operations 374
UpdateFile operations 378
UpdateFileType operations 382
UpdateGroup operations 384
UpdateRole operations 393
UpdateUser operations 398
User data type 515, 560
Volume data type 524
name element 421
Name parameter 96, 97, 182, 228
Name property 50, 120, 250
name value pairs 477, 493
NameList element
as search criteria 116
CopyFile operations 248
DeleteChannel operations 256
DeleteFile operations 258
DeleteFileType operations 259
DeleteGroup operations 260
DeleteRole operations 263
DeleteUser operations 264
MoveFile operations 330
SelectChannels operations 343
SelectFiles operations 345
SelectFileTypes operations 347
SelectGroups operations 348
SelectRoles operations 357
SelectUsers operations 359
UpdateChannel operations 374
UpdateFile operations 378, 608
UpdateFileType operations 382
UpdateGroup operations 384
UpdateRole operations 393
UpdateUser operations 398
NameList parameter 96, 97, 182, 228
names
See also user names
defining file types and driver 442

names *(continued)*
duplicating 18, 254, 329
getting parameter 483
getting volume 28, 321
localizing installations and 625
portType definitions and 9
retrieving translated role 326, 549
text string limits for 643
namespace attribute types 18, 219
namespace declarations 6, 17, 18, 218, 223, 225
NameValuePair class 201
NameValuePair data type 477
naming
channels 415
data types 8, 514
Encyclopedia volumes 524
files 346, 436
folders 251
iHubs 508, 509
log files 566, 567, 568
output files 270
parameters 434, 482
printers 492, 524
resource groups 496, 509
security roles 499
web services 10
naming collisions 18
naming-java.jar 575
native file types 105
.NET client sample application 222
.NET client solutions 216
NeverExpire element 410, 455, 605, 606
NewFile data type 477
NewFile element 105, 340, 404
NewFile objects 235
NewFile variable 191, 234
newUser method 179
NextStartTime element 466, 470
no-cache attribute 17
node start or stop errors 573
NodeLockViolation element 322, 508
NodeLockViolationExpirationDate element 508
nodes. *See* cluster nodes
NoEvent element 432
non-native reports. *See* third-party reports
Normal mode 147, 149, 334
notification groups
adding users 133, 385, 388
creating 365, 444
deleting 259
getting properties for 348
getting users in 347, 551, 552
removing users 385
scheduling jobs and 34, 389
searching 445, 446
sending e-mail to 56, 57, 58, 338, 365
updating 129, 383, 384, 385
notifications
addressing 57, 125
archiving 598
customizing 61
data transfer protocols for 4
deleting 261
directing to channels 57, 59, 60, 459
directing to external users 552, 556, 560
expiring 611, 612
getting properties for 306, 352
overriding recipient preferences for 59, 366, 456
overview 56
removing users 388
searching 459, 460
selecting 351
sending attachments with. *See* attachments
sending completion 339, 516, 561
sending large messages and 90
sending to multiple locales 61
setting expiration intervals for 516, 525, 561
setting number sent 452, 467
submitting jobs and 56, 57, 58, 351, 457
NotifiedChannelId element
GetNoticeJobDetails operations 307
JobNotice data type 459
JobNoticeSearch data type 461
JobScheduleSearch data type 472
JobSearch data type 473
NotifiedChannelName element
GetNoticeJobDetails operations 307
JobNotice data type 459
JobNoticeSearch data type 461
JobScheduleSearch data type 472

JobSearch data type 473
NotifiedUserId element 458, 461, 471, 473
NotifiedUserName element 459, 461, 471, 473
NotifyChannels element 62, 305, 308
NotifyChannels property 303, 307
NotifyChannelsById element 339, 366
NotifyChannelsByName element 338, 365
NotifyCount element 452, 467
NotifyGroups element 62, 304, 308
NotifyGroups property 303, 307
NotifyGroupsById element 339, 366
NotifyGroupsByName element 338, 365
NotifyUsers element 62, 304, 308
NotifyUsers property 303, 306
NotifyUsersById element 338, 365
NotifyUsersByName element 338, 365
null values 408, 482, 485
NumberOfCopies element 462, 490, 492
NumBytes element 148, 335
numeric values 419, 449

O

Object element
DataExtraction operations 255
GetContent operations 278
GetCustomFormatData operations 280
GetDynamicData operations 284
GetFormats operations 299
GetJavaReportEmbeddedComponent 300
GetJavaReportTOC operations 302
GetMetaData operations 305
GetPageCount operations 275, 309
GetPageNumbers operations 310
GetParameterPicklist operations 311
GetStaticData operations 316
GetStyleSheet operations 317
GetTOC operations 323
SaveTransientReport operations 340
SearchReport operations 341
SelectJavaReportPage operations 349
SelectPage operations 355
object IDs 47, 198, 479
object operation code 580
object operation descriptions 586
object renderer package 531
object type code 580
object type descriptions 586
ObjectId element
CancelReport operations 246
ExecuteQuery operations 268
ExecuteReport operations 198, 271
GetEmbeddedComponent operations 286
GetSyncJobInfo operations 318
OpenInfoObject operations 331
PendingSyncJob data type 486
RunningJobs data type 503
WaitForExecuteReport operations 405, 406
ObjectIdentifier data type 479
ObjectName element 331
ObjectOperationCode entry 580, 586
ObjectOperationDescription entry 586
objects
counting 89
getting custom formats for 280
getting properties for 185
ignoring duplicate 124
mapping 575
not finding 124
retrieving 89
searching for hidden 440
setting expiration intervals for 410
setting privileges for 556
updating multiple 124
viewing logging information about 586
ObjectTypeCode entry 580, 586
ObjectTypeDescription entry 586
ODBO API functions 330
ODBORequest element 330
ODBOResponse element 331
ODBOTunnel operations 330
ODBOTunnel type definition 330
ODBOTunnelResponse type definition 330
OFF logging level 531
Offline element 510
Offline value 511
ojdbc14.jar 575
OLAP servers 28, 330
OnceADay element 408, 423, 476, 526
OnDay element 476
Online Archive Driver 598–604
Online Archive Driver folder 598, 600
Online Archive option 598

online archive service 602
online archive service provider 602
online backup schedules 139
online documentation xvii
Online element 510
online help. *See* online documentation
Online value 511
onlinearchive.cfg 599, 600, 601
OnlineBackupSchedule property 139, 325
OnlineOnly element 319, 321
Open Security applications 359, 524
Open Security cache 391
Open Security library 244, 551
Open Security page 539
Open Security web service 538
open server applications 367
open server drivers
  pinging 28, 147, 148, 333
  setting time-out intervals for 479
open server options 270, 479
open server reports 35
open server technology 4, 46
open source logging tool 531
OpenInfoObject operations 35, 331
OpenInfoObject type definition 331
OpenInfoObjectResponse type definition 332
OpenSecuritySelectGroupsOfUser element 524
OpenSecuritySelectUsersOfRole element 524
OpenServerOptions data type 479
OpenServerOptions element 270, 367
Operand1 element 426
Operand2 element 426
Operand3 element 426
operating systems 14, 511
Operation element
  DataFilterCondition data type 426
  FilterCriteria data type 443
  GetEmbeddedComponent operations 286
  SubmitJob operations 54, 365
operation element 9
operation ID lists 96
operation IDs 96
operational information 570
operations
  See also specific type
  accessing channels and 29
  accessing iHub and 26
  applying ACLs and 114
  archiving files and 604
  binding to messages 5, 9, 10, 167, 221
  creating 9, 643
  defining composite 30, 124, 132
  defining data for 96–98
  defining web service 5, 10
  deleting items and 127
  displaying schema definitions for 11
  grouping administrative 124
  handling errors with 124
  logging usage information and 566
  managing report files and 31, 35
  managing users and 42
  managing volume data and 29, 181, 226
  performance monitoring and 594
  retrieving external users and 545
  retrieving multiple objects and 89
  running jobs and 35
  running multiple 14
  running queries and 35
  searching reports and 40
  sending notifications and 34
  transactions and 58, 132, 133
  viewing data and 76
  viewing logging information for 586
operations reference 239
Operator element 559
Operator role 433
operators 41, 443
optimizing performance 594
optional data 19
options 328
Options element 429
Oracle databases 575, 579
Orientation element 462, 490, 492
OrientationOptions element 490
OSVersion element 511
outdated reports 598
–output command line option 632
output
  converting 362
  creating silent installs and 628
  formatting. *See* output formats
  preserving 46

saving 198
sending notifications and 56, 59
output columns 419
Output element 552
output element 10
output file access types 305, 309
output file types 47, 271, 405, 442
output files
assigning to jobs 452, 466
changing archive rules for 605
creating job notifications and 366, 458, 459
defining attributes of 46
encrypting 632
formatting content 59, 419
getting access rights to 305
naming 270, 365
running silent installs and 634
saving 46, 270, 367, 457
sending as attachments 56, 366
specifying 54
updating archiving rules for 608
updating job schedules and 390
output format code 581, 586
output format descriptions 586
output formats
getting conversion options for 284
overriding 56, 366, 456
sending e-mail attachments and 59, 367, 456
sending third-party reports and 56
viewing logging information for 586
output message element 9
output messages
See also responses
defining 9, 10
printing 190, 233
specifying 167, 221
output parameters 388
OutputFileAccessType element 305, 309
OutputFileName element 458
OutputFileSize element 452, 466
OutputFileType element 268, 271, 405
OutputFileVersion element 459
OutputFileVersionName element 454, 458
OutputFormat element
DataExtractionFormat data type 425
DocumentConversionOptions data type 429
ExecuteReport operations 270
GetDocumentConversionOptions operations 284
SelectJavaReportPage operations 349
OutputFormatCode entry 581, 586
OutputFormatDescription entry 586
OutputMaxVersion element 454
OutputParameter element 245
OutputProperties element 342, 350
OutputType element 442
OutputType property 250
OverrideRecipientPref element 339, 366, 456
overriding
aging and archiving rules 604
notification preferences 59, 366, 456
output formats 56, 366, 456
report execution wait times 40
viewer preferences 40
oversize pages 79, 355
overwriting report files 105, 131, 330, 478
Owner element
File data type 436
FileInfo data type 619
FileSearch data type 440
JobProperties data type 451, 465, 470
JobSearch data type 473
PendingSyncJob data type 486
RunningJobs data type 503
owner information 599
Owner property 120, 298
OwnsVolume element 508

P

packages 530
page components 355
Page element 77, 349, 355
page heights 79, 355
Page Level Security option 530
page numbers 310, 480
page properties 78
page ranges 480
Page Security application 531, 532, 540, 541
page size attributes 79
page widths 79, 355

Page_Security_Example directory 541
PageCount element 309, 436, 466, 619
PageCount property 120, 298
PageHeight element 355
PageIdentifier data type 480
page-level security 530, 541, 543
page-level security application. *See* Page Security application
PageNum element 77, 480
PageNumber element 310
PageRange element 463
PageRef element 78, 350, 356
pages
  counting 87, 275, 309
  retrieving 200, 348, 354
  searching for 354, 494
  setting range of 77, 463
  setting size 79, 462, 490, 492
PageSecureViewing value 328
PageSize element 462, 490, 492
PageSizeOptions element 490
PageWidth element 355
paging order 89
paper orientation 462, 490, 492
paper size 79, 320, 463
paper trays 462, 491, 492
PaperLength element 463
PaperTray element 462, 491, 492
PaperTrayOptions element 491
PaperWidth element 463
parameter definitions 143, 273, 313, 480
parameter groups 482, 485
parameter lists 434, 483
parameter names 483
parameter pick lists 310, 312
parameter template files 639, 640
parameter types 495
parameter values files 38
  See also data object value files; report object value files
ParameterDefinition data type 480
ParameterDefinition element 113, 143
ParameterFile element 252
ParameterFileId element 466, 470
ParameterFileName element 466
ParameterList element 273, 297, 313
ParameterName element 311
ParameterPickList element 311
parameters
  assigning data types to 434, 482
  assigning values to 482, 484
  configuring custom event web service 206
  creating silent installs and 628, 632, 639
  defining control type attributes for 434, 483
  defining help text for 482
  defining report 480
  defining scalar 433, 504
  defining table 435
  enabling auto collection for 443
  encrypting passwords and 632
  exporting 144, 273
  generating locale-specific data and 5
  getting database connection 281
  getting definitions for 143, 273, 313
  getting file type 139, 142, 297
  getting values of 51, 312
  logging error information and 573, 593
  logging usage information and 592
  naming 434, 482
  prompting for 485
  running reports and 50, 54
  search operations and 83, 182, 185, 227
  selecting 485
  setting name-value pairs for 46
  submitting job requests and 365, 390
  testing for 54
  uploading files and 192, 193
  viewing Reportlets and 279
  writing to files 252
  writing to information objects 427, 484, 485
ParameterValue data type 484
ParameterValueFileId element 365
ParameterValueFileName element 365
ParameterValueList element 252
ParameterValues element 50, 270, 365
ParameterValuesFile element 253
ParameterValuesFileId element 270
ParameterValuesFileName element 270
parent role 395, 501
ParentRoleId element 501
ParentRoleName element 501
parser 19

partition names 148
PartitionName element 148, 335
partitions 146, 272
PassThrough message 42, 244
PassThrough operations 551
pass-through security 548
PassThrough type definition 551
PassThroughResponse type definition 552
Password element 327, 516, 546
Password property 336
password variable 173, 223
PasswordCtl element 633
passwords
  changing 516
  creating silent installs and 632, 633
  creating system 368
  creating user 327, 516
  encrypting 632
  external authentication and 530, 548
  logging in to Encyclopedia and 99, 176
  logging in to iHub Systems and 101
  requiring 482
  specifying default 632
  starting logging consolidator and 579
PathInformation element 522
paths
  creating folders and 251
  downloading files and 111
  getting display formats and 87
  getting page counts and 87
  installation 623, 635, 641
  searching folders and 121
  setting error log 593
  setting usage log 592
  uploading files and 107
Payload element 148, 336, 337
PDF formats 80, 521
PDF reports 79
PdfQuality element 523
Pending element 319, 432
pending job attributes 290
pending job status 432
pending jobs 54, 152, 155, 289, 485
pending requests 49
  See also pending jobs
Pending status message 49
Pending value 138, 156
PendingSyncJob data type 485
PendingSyncJobs element 288, 292
PendingSyncJobsResultDef element 153, 290
PendingTime element 157, 487
PercentTransientReportCacheInUse element 289
perfmon utility 594
performance counters 594, 595
Performance Monitoring API 594
performance monitoring operations 594
Permission data type 487, 558, 619
Permission element 115, 617
Permissions property 106, 404
permissions. *See* privileges
persistent connections 108
persistent reports 46, 47, 54, 198, 448
personal channels 57
physical address (SOAP port) 169
pick lists 310, 312
Ping element 149, 150
Ping operations 332
Ping requests
  creating 146
  returning diagnostic information and 147
  sending 149, 150
  setting payload length for 148
Ping responses 147, 148
Ping type definition 332
PingResponse element 149, 150, 151
PingResponse type definition 336
plain text messages 61
platform-independent reporting 4, 14, 93
plug-ins 16
PMD. *See* Process Management Daemon
polling parameters 207
PollingDuration element 430
PollingInterval element 415, 430, 455, 480
port names 10, 167
ports 11, 169, 207, 475
portType definitions 9, 10
portType element 6, 9, 167, 221
Position element 482, 485
POST directive 16
PostgreSQL database 633, 635, 641
PostgreSQLDatabaseProfile element 633
PostgreSQLServiceProfile element 633
PostResponseRef element 279, 356

PPT formats 80, 521
Pragma directive 17
PrecedingParameterValues element 311
preferences (users). *See* user preferences
preferences (viewer) 99, 524, 556
prepareDownloadFileCall method 196
primary keys 580
print jobs
  cancelling 64
  creating 337
  preserving status information for 367
  prioritizing 556
  retrying 339, 498
  running immediately 365
  scheduling 54, 338, 365
  setting printer options for 54, 461
  setting priority for 338
print requests. *See* print jobs
print to file property 463
Printer data type 488, 644
printer options 54, 338, 461, 491
printer settings. *See* printer options
PrinterName element 320, 325, 462, 492
PrinterOptions data type 491
PrinterOptions element
  GetJobDetails operations 62, 304
  GetNoticeJobDetails operations 308
  GetUserLicenseOptions operations 324
  GetUserPrinterOptions operations 325
  GetVolumeProperties operations 326
  PrintReport operations 338
  SubmitJob operations 365
PrinterOptions property 139, 303, 306, 325
printers
  assigning to Encyclopedia 403
  defining system 488
  getting information about 139, 140, 320
  getting settings for 139, 142, 324
  naming 492, 524
  setting options for 54, 338, 461, 491
  specifying default 492
  updating options for 403
  updating user list for 401
Printers element 320
printing
  oversize pages 355
  reports 46, 54, 78, 337
  to files 463
printing events 569, 571
printing requests. *See* print jobs
PrintReport operations 39, 337
PrintReport type definition 337
PrintReportResponse type definition 339
PrintToFile element 463
printUsage variable 173, 223
prioritizing jobs 66, 129, 338, 364
Priority element 338, 364, 452, 465, 470
Private element 437
private files 105, 118, 437, 617
Private value 436
privilege attributes 488
privilege filters 117, 119
PrivilegeFilter data type 493
PrivilegeFilter element 120, 417, 440
privileges
  accessing report files and 115, 116, 377
  archiving reports and 619
  assigning 487
  changing 114, 380, 401
  filtering 493
  retrieving 102, 103, 104
  searching for 417, 440
  sending output files and 57
  setting on report executables 544
  subscribing to channels and 130, 374, 375, 415
  testing RSSE sample applications and 540
  updating 112, 114, 390
  validating external users and 558
process IDs 504
Process Management Daemon 574
ProcessID element 148, 335
ProgramFolderCtl element 630
programming environments 9
programming languages 14
progressive viewing 46, 48, 49, 270, 503
ProgressiveViewing element 267, 270
PromptFilterCriteria element 444
PromptParameter element 485
prompts 638
PROP_DOMULTIREFS parameter 193
properties
  archive service provider 603
  channels 130, 373

connections 277, 360, 548, 557
Encyclopedia volumes 125, 325, 402, 403
error log files 566, 568
file copy operations 106, 107
file selection operations 116
file update operations 112, 377
file upload operations 105, 404
folders 116, 123, 377
get file details operations 123, 296
get folder item operations 120, 121
iHub 151
jobs
getting 61, 63, 137, 152, 302, 306
setting 463
notification groups 348
notifications 306, 352
Open Security applications 539
output 362
resource groups
changing 68
getting 69, 70, 314
setting 65, 71, 361
updating 68, 392
RSSE applications 556, 559
search results 185, 342, 350
security roles 357, 393
select page operations 78, 201
usage log files 565, 567
users 397, 398, 401, 550
Properties element 255
property files 531
property name-value definitions 493, 559
PropertyValue data type 493, 559
ProviderName element 617
proxy classes 5
proxy DLLs 216
proxy namespaces 225
proxy objects 14, 223
proxy servers 168, 174, 216
PurgeUserInfo element 264
purging errors 573

Q

queries
adding columns to 418
filtering data with 255, 443
getting content of 79
grouping data with 446
running 266
setting sort order for 512
updating parameters for 390
Query element
ExecuteQuery operations 267
GetJobDetails operations 62, 305
GetNoticeJobDetails operations 308
OpenInfoObject operations 331
Query event type 569
query operators 443
query status attributes 268
query types 513
QueryFileId element 267
QueryFileName element 267
QueryPattern element 552, 553, 554
QueuePosition element 487
QueueTimeout element 157, 487

R

radio buttons 483
Range data type 494
Range element 84, 342, 480
read privilege 488, 620
ReadFile requests 334
ReadFile value 147
Reason element 635
RebootRequired element 635
reboots 635
Record data type 494
RecordDefinition element 483
RecordFailureStatus element 339, 367, 456
records 89, 417
See also rows
RecordSuccessStatus element 339, 367, 456
recurring jobs 423, 476, 494
recurring reports 526
Recursive element
CopyFile operations 248
DeleteFile operations 258
GetJavaReportTOC operations 302
MoveFile operations 329
SelectFiles operations 345
UpdateFile operations 116, 378
recursive operations 114, 116, 127

RedirectPath element 287, 523
References.cs 225
refresh intervals 576
registry keys 622, 623
relative paths 107
remote procedure calls 164, 168, 219, 221
remote services 168, 221
RemoteException parameter 194
RemoveAll InstallShield parameter 636
RemoveArchiveRules element 380
RemoveArchiveRules operations 605
RemoveChannelNotificationById element 390
RemoveChannelNotificationByName element 389
RemoveChildRolesById element 396
RemoveChildRolesByName element 395
RemoveDependentFilesById element 380
RemoveDependentFilesByName element 379
RemoveFileCreationPermissions element 401
RemoveFromGroupsById element 400
RemoveFromGroupsByName element 400
RemoveGroupNotificationById element 389
RemoveGroupNotificationByName element 389
RemoveLicenseOptions element 401
RemoveOutputFilePermissions element 390
RemoveParentRolesById element 396
RemoveParentRolesByName element 395
RemoveRequiredFilesById element 380
RemoveRequiredFilesByName element 380
RemoveSubscribersById element 375
RemoveSubscribersByName element 375
RemoveUserNotificationById element 389
RemoveUserNotificationByName element 388
RemoveUsersById element 385
RemoveUsersByName element 385
removing. *See* deleting
renderer package 531
Repeat data type 494
Repeat element 408, 423, 476, 526
ReplaceExisting element 105, 248, 330, 478
ReplaceLatestDropDependency element 520
ReplaceLatestDropDependency value 478
ReplaceLatestIfNoDependents element 520
ReplaceLatestIfNoDependents value 478
ReplaceLatestMigrateDependency element 520
ReplaceLatestMigrateDependency value 478
ReplaceLatestVersion element 454
Reply element 336
report components 278, 422
  See also components; reports
report design files 106
report documents. *See* documents; reports
Report Encyclopedia. *See* Encyclopedia volumes
report execution requests. *See* ExecuteReport operations
report files
  See also specific report file type
  applying ACLs to 114
  archiving 112, 604, 606, 607, 610, 613
  attaching to e-mail messages 56, 92, 438
  attaching to SOAP requests or responses 108, 111
  bundling 268, 270, 367, 454
  changing properties for 377
  copying 131, 247, 599
  copying properties of 107
  creating 131, 477
  defining attributes of 435
  deleting 127, 257, 598, 614
  determining if private or shared 617
  downloading 90, 111, 112, 194, 235
  embedding 91, 108, 194
  encrypting 632
  exporting 442
  getting access rights to 103
  getting ACLs for 101, 292
  getting expired 615
  getting privileges for 103, 104
  getting properties for 116, 123, 296
  monitoring 438
  moving 131, 328
  naming. *See* file names
  overwriting 105, 131, 330, 478
  programming tasks for 31
  reading 192
  returning information about 344, 618
  returning list of 117, 120, 344
  returning location of 297

running or printing 53
searching for 116, 118, 346, 437, 439
selecting 116, 344
setting expiration policy for 598, 603
setting privileges for 114, 116, 380, 390
specifying type 240
updating 112, 129, 377, 378, 381
updating parameters for 113
uploading 90, 105, 109, 190, 233, 404
versioning options for 105, 519
report generation events 570
report generation requests
assigning to resource groups 21, 240
cancelling 49, 64, 156, 157, 405, 414
creating 46, 269
prioritizing 556
retrying 498
scheduling 54
setting status of 432
setting wait intervals for 48, 49, 268, 270
submitting jobs for 365
report IDs 505
report object value files
creating 252, 270
exporting parameter definitions in 273
generating reports from 365
retrieving parameters in 51, 143, 273, 312
saving 270
sending as attachments 144
report objects. *See* reports
report parameters. *See* parameters
Report Server Security Extension 530
See also RSSE applications; RSSE API
report servers. *See* iHub; servers
report status attributes 246, 271
ReportDeletion event type 569
ReportFileId element 312
ReportFileName element 312
ReportGeneration event type 569, 570
ReportGeneration value 328
reporting system. *See* iHub System
Reportlet format 80, 521
Reportlet parameters 279
Reportlets 80, 279, 522
ReportParameters element 62, 305, 308
ReportParameters property 303, 307
ReportParameterType data type 495
ReportParameterType element 312
ReportPrinting event type 569, 571
reports
assigning to resource groups 73
building security applications for 530, 541
cancelling 246
controlling access to 558
converting output for 361
counting number of pages in 87, 275, 309
customizing 77
deploying 543–545
displaying 76, 77, 201, 520
embedding files in 438
generating 46, 54, 198
getting contents of 79, 278
getting currently running 152
getting custom formats for 88, 280
getting dynamic data from 284
getting embedded components in 285
getting specific pages in 77, 354
getting status of 47, 432
getting style sheets for 317
getting supported formats for 87
loading RSSE sample 540
localizing 5
managing third-party 4
monitoring 156
printing 46, 54, 78, 337
running 46, 50, 206, 269, 405
saving output for 46
saving temporary 340
searching 40, 82, 341, 505, 506
splitting across multiple pages 79
submitting job requests for 53, 361
uploading 106, 107, 543
viewing first or last page of 77
ReportType element 240, 343, 496
ReportViewing event type 569, 571
repository. *See* Encyclopedia volumes
Request element 512
RequestConnectionHandle element 330
RequestedHeadline element 466
RequestedOutputFile element 267, 270, 365
RequestedOutputFileName element 466, 471, 473
RequestID element 21, 240, 449
request-response message pairs 9, 14, 177

requests
See also specific operation request
ad hoc parameters and 143
adding authentication IDs for 99
adding time stamps to 121
administering Encyclopedia and 178, 182, 187, 226, 227, 231
archiving and 610, 614, 616
attaching files to 108
caching directive for 17
creating message definitions for 9
creating resource groups and 66
defining 9, 21
deleting data and 98
directing to BIRT iHub 14, 18
downloading files and 111, 112
duplicating 124, 249
embedding files in 91, 108
executing reports and 47, 49
extending wait times for 49
failing 22
forwarding to targets 17
grouping transactions and 30
identifying 21
initiating 16
locale-specific reports and 5, 93
recursive search operations and 116
required elements of 19, 448
retrieving multiple objects and 89
routing to alternate MDS 319
search operations and 82, 84
sending 21, 177, 178, 240
specifying output file names for 54
testing reporting environment and 28
uploading files and 109, 110
required data 19
required files 440
required parameters 54, 434, 482
required passwords 482
RequiredFileId element 440
RequiredFileName element 440
Reserved element 497
Reserved property 392
reserved resource groups 65, 392, 497
ResetCounters operations 596
ResetCounters type definition 596
ResetCountersResponse type definition 596
Resolution element 462, 490, 492
ResolutionOptions element 490
resource group names 75
resource group settings 497
resource groups
assigning to
BIRT iHub 66, 71, 509
Encyclopedia volumes 66
jobs 65, 74, 364, 465
reports 73
requests 21, 240
changing properties of 68
configuring 28
creating 26, 65, 66, 253, 495
deleting 26, 73, 262
enabling or disabling 66
getting information about 27, 313, 315
getting job-specific 75
getting list of 27, 69, 314
getting properties for 69, 70
naming 496, 509
overview 65
reserving 67
running jobs and 496, 509
setting priority for 496
setting properties for 65, 71, 361
specifying as reserved 65
specifying asynchronous 66
specifying synchronous 67
updating 361, 392
viewing logging information for 587
ResourceGroup data type 495
ResourceGroup element
CreateResourceGroup operations 253
GetResourceGroupInfo operations 313
JobProperties data type 465
RunningJob data type 504
SubmitJob operations 364
UpdateResourceGroup operations 392
ResourceGroup property 303, 307
ResourceGroupId entry 587
ResourceGroupName element 509
ResourceGroupName entry 587
ResourceGroupSettings data type 497
ResourceGroupSettingsList element 253, 314, 392
ResourcePath element 525

resources 564, 567, 626
responses
See also specific operation response
Administrate operations and 124
archiving and 607, 610, 611
attaching files to 108, 111, 270
creating message definitions for 9
creating multilingual 93
defining 9, 21
disabling caching for 17
downloading files and 112
embedding files in 91, 108, 194
executing reports and 48
grouping transactions and 30
locale-specific reports and 93
omitting 9
retrieving multiple objects and 89
retrieving printer information and 320, 324
retrieving report content and 79, 278
selecting report pages and 354
sending attachments and 90, 92
setting wait intervals for 48, 270
specifying media types for 16
testing report completion and 87
viewing specific pages and 77, 78
restricting access to Encyclopedia 98
Result element 635
result set schemas 305, 498
result sets 89, 519
See also queries; search results
ResultDef element
GetFileDetails operations 296
GetFolderItems operations 120, 298
GetJobDetails operations 303
GetNextExpiredFiles operations 615
GetNoticeJobDetails operations 306
GetUserProperties operations 550
GetVolumeProperties operations 139, 325
SelectChannels operations 343
SelectFiles operations 117, 345
SelectFileTypes operations 347
SelectGroups operations 348
SelectJobNotices operations 352
SelectJobs operations 137, 351
SelectJobSchedules operations 353
SelectRoles operations 357
SelectUsers operations 358
ResultDef parameter 185
ResultDef String array 185
ResultSetName element 498
ResultSetSchema data type 498
ResultSetSchema element 255
RetainOwner file attribute 599
RetainPermission file attribute 599
RetainTimestamp file attribute 599
retry intervals 499
retry options 339, 498
RetryInterval element 455, 499
RetryOption element 454, 499
RetryOptions data type 498
RetryOptions element 339, 367
RetryOptionType data type 499
ReturnCode element 245, 552
ReturnDataInBlocks element 331
RevokePermissions element 375, 380
rich text formats. *See* RTF formats
Role data type 499, 645
Role element 253
role names 499, 500, 549
RoleCondition data type 500
RoleField data type 500
RoleId element 360, 487
RoleName element
DoesRoleExist operations 547
Permission data type 487, 558, 619
SelectUsers operations 554
SetConnectionProperties operations 360
RoleNames element 277
roles
adding to Login requests 99
administering Encyclopedia and 30
archiving 599
assigning privileges 487, 619
creating 126, 253, 433, 499, 500
deleting 262, 395
duplicating 254
getting privileges for 102, 104
getting properties for 357
getting volume-specific 139
mapping external users to 549, 559
naming 499
searching 356, 500, 553
setting filters for 117, 119

roles *(continued)*
setting properties for 556
testing for external 547
testing reporting environment and 28
translating 549
updating 98, 393, 394, 396, 400
validating 99, 327
Roles element 358, 554
RoleSearch data type 500
rolling back upgrade installations 629, 638
RoutedToNode element 452, 466
.rov files. *See* report object value files
rows 426, 449
See also records
RPT files. *See* Crystal reports
RPX files. *See* Crystal reports
RSSE API 41
RSSE API library 244
RSSE API reference 531
RSSE applications
See also Report Server Security Extension
accessing sample 530
building 532, 541
configuring logging levels for 531
enabling web services for 538
initializing 555
installing 531, 532
migrating 540
running as web services 530, 532, 533
specifying credentials for 327
stopping 557
translating access control lists and 541
RSSE data types 557
RSSE external registration 357, 358
RSSE interface 530
RSSE objects 531
RSSE operations 545
RSSE service 541
rsse.jar 531
rsseAcl.jar 532
rsseAuthenticate.jar 532
rsseLdap.jar 532
RSSEVersion element 556
RTF formats 61, 80, 521
rules. *See* aging rules; archiving rules
run requests. *See* report generation requests
runAdminOperation method 181, 189
RunAndPrintReport value 54
RunAsUser element 327
RunLatestVersion element 54, 267, 364, 466
running
jobs 52, 54, 70, 408, 502
queries 266
reports 46, 50, 206, 269, 405
sample applications 170, 222
silent installs 633, 634, 638
silent uninstalls 635, 636, 641
Running element 319
running job attributes 291
Running value 138, 156
RunningJobs data type 502
RunningJobs element 289, 292
RunningJobsResultDef element 153, 291
RunningSyncJobs element 289
RunningTime element 504
RunOn element 408, 476, 526
RunReport value 54
run-time generation requests 21, 240

S

–s command line option 640, 641
sample applications
accessing code libraries for 163
accessing Java RSSE 530
building Java RSSE 532, 541
logging in to Encyclopedia and 172, 176
running 170, 222
sample custom event web service 206, 209, 210
sample package 209
sample projects 216
sample reports 540, 543, 630
SampleEventService class 209, 210
SaveOutputFile element 267, 270
SaveSearch operations 41, 339
SaveSearch type definition 339
SaveSearchResponse type definition 340
SaveTransientReport operations 33, 340
SaveTransientReport type definition 340
SaveTransientReportResponse type definition 341
saving
output files 46, 270, 367, 457

report object value files 270
report output 198
search results 339
temporary reports 340
scalar parameters 433, 504
ScalarDataType data type 504
Scale element 462, 490, 492
ScaleOptions element 490
scaling factor (printer) 462, 490
scaling options 490
ScalingFactor element 286, 522
schedule attributes 408
schedule type attributes 469
scheduled jobs 262, 353
See also jobs
ScheduleDetails element 467
ScheduleEndDate element 469
schedules
changing properties for 129
creating job 423, 467, 468, 475, 525
getting archiving 139
getting information about 353
getting online backup 139
searching 467, 469, 470
setting archiving 403, 606, 609
setting report generation 54, 206, 365, 606
updating 129, 386, 387, 391
Schedules element
GetJobDetails operations 61, 304
GetNoticeJobDetails operations 308
PrintReport operations 338
SubmitJob operations 365
Schedules property 303, 306
ScheduleStartDate element 469
ScheduleType element 469
scheduling reports 54, 206, 365, 606
schemas 9, 305, 411, 427, 498, 580
See also WSDL schemas
schemas classes 172, 199
schemas library 162
schemas package 162, 173
scripts 575, 578
SDI. *See* Service Definition Interface
search conditions
defining parameters for 182, 185, 227
entering special characters in 416, 437, 445
entering wildcards in 183, 228
moving files and 329
retrieving files and 116, 118, 122, 184, 229
retrieving security roles and 357
setting 40, 82, 182, 227
specifying multiple 184, 229
search criteria. *See* search conditions
Search element
CopyFile operations 248
defining as empty 114, 116
DeleteChannel operations 256
DeleteFile operations 258
DeleteGroup operations 97, 259
DeleteJob operations 260, 262
DeleteJobNotices operations 261
DeleteRole operations 263
DeleteUser operations 264
GetFolderItems operations 298
MoveFile operations 329
SelectChannels operations 343
SelectFiles operations 116, 118, 345
SelectFileTypes operations 347
SelectGroups operations 348
SelectJobNotices operations 352
SelectJobs operations 351
SelectJobSchedules operations 353
SelectRoles operations 357
SelectUsers operations 358
UpdateChannel operations 373
UpdateFile operations 378
UpdateGroup operations 384
UpdateJobSchedule operations 386
UpdateRole operations 393
UpdateUser operations 397
Search event type 569, 571
search events 571
search field attributes 416
search fields 445, 451, 460
search formats 299, 444
search lists 506
search operations
See also searching
constructing composite messages for 41
copying files and 132
creating 82, 84, 182, 227
defining as recursive 116
described 136
external security systems and 41

search operations *(continued)*
limiting scope of 97
moving files and 131
programming tasks for 40
setting conditions for. *See* search conditions
search operators 41
Search parameter 97, 182, 228
search results
defining attributes for 507
displaying 83
exceeding limits of 89, 344
getting 315
retrieving items from 276
retrieving number of 359
saving 339
setting CSV options for 342, 350
setting range of pages for 84
search viewing parameters 341
searchByCondition method 183
SearchByIdList value 505
SearchByNameList value 506
SearchCriteria element 355
SearchFile element 340
searching
See also search operations
channels 343, 415, 416
CSV files 342, 350
directories 116
Encyclopedia volumes 116, 118, 136, 182, 227
external users 554
files 116, 118, 437, 439
folders 116, 120, 122
for specific components 82
for specific data 97
groups 347, 445, 446
hidden objects and 440
jobs 450, 472
notifications 459, 460
range of pages 84, 354, 494
reports 40, 82, 341, 505, 506
schedules 467, 469, 470
security roles 356, 500, 553
specific file types 346
subfolders 116, 248, 302, 329, 345
users 358, 517, 518
SearchList element 315, 340
SearchList value 506
SearchObject element 315
SearchRef element 342
SearchReport element 83, 84
SearchReport operations 41, 82, 84, 341, 522
SearchReport type definition 341
SearchReportByIdList data type 505
SearchReportByIdList element 505
SearchReportByIdNameList data type 506
SearchReportByIdNameList element 506
SearchReportByNameList data type 506
SearchReportByNameList element 506
SearchReportResponse element 85
SearchReportResponse type definition 342
SearchResultProperty data type 507
secure read privilege 488, 620
security applications 530, 541, 543
security integration options 524
security operations 42, 98
security roles
adding to Login requests 99
administering Encyclopedia and 30
archiving 599
assigning privileges 487, 619
creating 126, 253, 433, 499, 500
deleting 262, 395
duplicating 254
getting privileges for 102, 104
getting properties for 357
getting volume-specific 139
mapping external users to 549, 559
naming 499
searching 356, 500, 553
setting filters for 117, 119
setting properties for 556
testing for external 547
testing reporting environment and 28
translating 549
updating 98, 393, 394, 396, 400
validating 99, 327
SecurityIntegrationOption element 524
Select operations 136
SelectByIdList value 505
SelectByNameList value 506
SelectChannels operations 343
SelectChannels type definition 343

SelectChannelsResponse type definition 344
SelectFiles element 117, 120
selectFiles method 183, 184, 185, 229
SelectFiles objects 185
SelectFiles operations
  defining 344
  returning properties and 116–120, 182, 228
  searching and 41
SelectFiles type definition 344
SelectFilesResponse element 118, 119
SelectFilesResponse objects 184, 229
SelectFilesResponse type definition 346
SelectFileTypes operations 41, 346
SelectFileTypes type definition 346
SelectFileTypesResponse type definition 347
SelectGroups operations 34, 347, 552
SelectGroups type definition 347, 552
SelectGroupsOfUser element 557
SelectGroupsResponse type definition 348, 552
selection lists. *See* pick lists
SelectJavaReportPage class 201
selectJavaReportPage method 201
SelectJavaReportPage operations 39, 348
SelectJavaReportPage type definition 348
SelectJavaReportPageResponse type definition 350
SelectJobNotices operations 39, 351
SelectJobNotices type definition 352
SelectJobNoticesResponse type definition 352
SelectJobs element 138
SelectJobs operations 39, 137, 350
SelectJobs type definition 351
SelectJobSchedules operations 39, 353
SelectJobSchedules type definition 353
SelectJobSchedulesResponse type definition 353
SelectJobsResponse element 138, 351
SelectList element 315, 340
SelectList value 506
SelectNameValueList element 434, 483
SelectPage element 77, 79
SelectPage operations
  printing and 78
  retrieving specific pages and 354
  viewing reports and 39, 76, 77
SelectPage type definition 354
SelectPageResponse element 78
SelectPageResponse type definition 356
SelectRoles method 553
SelectRoles operations 43, 356, 553
SelectRoles type definition 356, 553
SelectRolesOfUser method 553
SelectRolesResponse type definition 358, 553
SelectUsers element 137
SelectUsers operations 43, 136, 358, 554
SelectUsers type definition 358, 554
SelectUsersOfRole element 557
SelectUsersResponse element 137
SelectUsersResponse type definition 359, 554
SelectValueList element 434, 483
semicolon (;) character 54
SEND_TYPE_ATTR parameter 193
SendEmailForFailure element
  JobInputDetail data type 455
  PrintReport operations 339
  SubmitJob operations 366
  User data type 517, 561
SendEmailForSuccess element
  JobInputDetail data type 455
  PrintReport operations 339
  SubmitJob operations 366
  User data type 516, 561
SendFailureNotice element 339, 366, 455
sendmail utility 4
SendNoticeForFailure element 516, 561
SendNoticeForSuccess element 516, 561
SendSuccessNotice element 339, 366, 455
sequence element 8
ser package 166
Serialization class 220
serializing SOAP messages 16
server configuration error messages 574
Server element 148, 334
server home argument 578
Server Proxy.sln 216
server state attributes 511
server status attributes 508
ServerBuild element 511
ServerHome element 555
ServerInformation data type 507
ServerList element 321

ServerName element
GetFactoryServiceInfo operations 288
GetServerResourceGroupConfiguration operations 315
MDSInfo data type 475
PendingSyncJob data type 486
RunningJobs data type 503
ServerInformation data type 508
SetServerResourceGroupConfiguration operations 361
ServerResourceGroupSetting data type 509
ServerResourceGroupSettingList element 316, 361
servers
See also iHub
changing configurations for 508
creating log consolidator databases and 579
failing 511
getting default Encyclopedia volumes for 322
getting information about 151
getting list of 320
managing Encyclopedia volumes on 4
timing out 48
viewing error messages for 574
ServerState data type 510
ServerState element 475, 511
ServerStatusInformation data type 510
ServerStatusInformation element 508
serverURL variable 173, 223
ServerVersion element 152, 511
ServerVersionInformation data type 511
ServerVersionInformation element 322, 508
Service data type 512
Service Definition Interface 167
service definitions 10–11
service element 6, 10
service enable or disable errors 573
service type code 584
service type descriptions 587
ServiceList element 508
ServiceOrPort property 336
ServiceProfile element 633
services
See also specific iHub service; web services
calling remote 168, 221
developing web-based 4, 5, 9
event-based scheduling and 206, 209, 210
getting information about 152, 155, 288
page-level security and 540, 541
running RSSE applications as 530, 532, 533
viewing logging information for 579, 587
ServiceTypeCode entry 584, 587
ServiceTypeDescription entry 587
servlets 16
SessionID element 614, 615, 616
setAdminOperation method 181
SetArchiveRules element 381, 607
SetArchiveRules operations 605, 606
SetAttributes element
UpdateChannel element and 130
UpdateChannelOperation operations 375
UpdateFileOperation operations 379
UpdateFileTypeOperationGroup operations 382
UpdateGroupOperation operations 385
UpdateJobSchedule element and 129
UpdateJobScheduleOperation operations 388
UpdateRoleOperation operations 395
UpdateUserOperation operations 400
UpdateVolumePropertiesOperation operations 403
setAuthId method 177
SetAutoArchiveSchedules element 403, 609
SetAutoArchiveSchedules operations 606
SetBearingUsersById element 396
SetBearingUsersByName element 395
SetChannelNotificationById element 60, 390
SetChannelNotificationByName element 60, 389
SetChannelSubscriptionsById element 401
SetChannelSubscriptionsByName element 400
SetChildRolesById element 396
SetChildRolesByName element 395
setClassPath shell script 171
setClassPath.bat 170
SetConnectionProperties operations 43, 360
SetConnectionProperties type definition 360
SetConnectionPropertiesResponse type definition 361
setCreateUser method 181

SetDependentFilesById element 380
SetDependentFilesByName element 379
SetFileCreationPermissions element 401
SetGroupMembershipsById element 400
SetGroupMembershipsByName element 400
SetGroupNotificationById element 390
SetGroupNotificationByName element 389
setIgnoreDup method 180
SetLargeWebIcon element 383
SetLicenseOptions element 401
setOperationName method 194
setOperationStyle method 193
SetOutputFileAccess element 390
SetOutputFilePermissions element 390
SetParameterDefinitions element 113, 381, 382
SetParameterDefinitions operations 113
SetParameters element 129, 388
SetParameters operations 605
SetParameterValues element 390
SetParentRolesById element 396
SetParentRolesByName element 396
SetPermissions element 114, 130, 375, 380
SetPermissions operations 114
SetPrinterOptions element 388, 401, 403
SetQuery element 390
SetRequiredFilesById element 380
SetRequiredFilesByName element 380
SetRolesById element 401
SetRolesByName element 400
SetSchedules element 129, 388
SetServerResourceGroupConfiguration element 72
SetServerResourceGroupConfiguration operations 71, 361
SetServerResourceGroupConfiguration Response type definition 361
SetServerResourceGroupConfiguration type definition 361
SetSmallWebIcon element 383
SetSubscribersById element 375
SetSubscribersByName element 375
SetSystemPrinters element 403
settings. *See* properties
SetupTypeCtl element 630
SetupTypeDlg element 631
setUser method 180
SetUserNotificationById element 389
SetUserNotificationByName element 389
SetUsersById element 385
SetUsersByName element 385
SetWaitForEvent element 390
SetWindowsIcon element 382
severe logging level 572
severity levels (error logs) 572
Shared element 437
shared files 105, 118, 437, 617
shared libraries 564
Shared value 436
shell scripts 638, 639
ShortDescription element 442, 474
ShowInReportlet property 80
silent installations
  adding license file locations to 631
  creating 628, 638, 639
  customizing 629
  defining version and copyright information for 630
  encrypting passwords for 632
  hiding dialog boxes for 631
  running 633, 634, 638
  verifying 635
silent uninstalls 635, 636, 641
simple object access protocol. *See* SOAP
Size element 436, 619
Size property 120, 298
SkipPermissionError element 374, 398
small icons 383, 415, 442
SmallImageURL element 415, 442
SMTP messaging protocol 4
SOAP archiving data types 617
SOAP archiving operations 613, 614
SOAP endpoints 14
SOAP engine error messages 574
SOAP envelopes 14, 15, 17–18
SOAP header element 240
SOAP header extensions 174
SOAP headers
  adding authentication IDs to 99
  adding multilingual functionality to 93
  adding namespace declarations to 18
  binding definitions and 10
  creating 19–21
  defining attributes of 448

SOAP headers *(continued)*
overview 240
targeting resource groups and 65, 73
uploading files and 105
SOAP message IDs 449
SOAP messaging 4, 14
See also messages
soap namespace prefix 7
SOAP ports 10, 11, 169, 207
SOAP processor 162, 164, 216, 219
SOAP requests. *See* requests
SOAP responses. *See* responses
SOAP RSSE applications 532
SOAP RSSE data types 557
SOAP RSSE operations 545
SOAP VersionMismatch error 18
SOAP web service operations 212
SOAP with Attachments API 164
SOAPAction directive 17
soapAction element 10
soapAction parameter 193
sort columns 255, 427, 512
sort fields. *See* sort columns
sort order 446, 513
sort order attributes 427
SortColumn data type 512
SortColumnList element 255
SortDirection element 427
SortOrder element 513
source code
archive driver and 598
compiling 163
configuring RSSE logging levels and 531
creating LDAP configuration files and 533, 534
custom event web services and 209
generating 163
log consolidator application 576
logging extensions and 564, 567, 568
SourceForge.net code libraries 164
special characters 41, 416, 437, 445
splash screens 624, 625, 628
SplitOversizePages element 355
spreadsheet reports 200, 361
SQL databases 282
SQL script files 575
SQL statements. *See* queries
Standalone element 514
stand-alone servers 320, 321, 514
Standard logging level 566, 573
Start element 494
Start menu 630
Start operations 555
Start type definition 555
start_consolidator.sh 578
StartArchive command 145, 272, 610
StartArchive operations 616
StartArchive type 616
StartArchiveResponse type 617
StartArguments element 497, 498, 510
StartIndex element 311
starting
log consolidator application 576, 578, 579
RSSE applications 555
TCPMonitor utility 177
starting dates 469
Starting element 510
Starting value 511
StartPartitionPhaseOut command 145, 272
StartResponse type definition 555
StartRowNumber element 255
StartTime element 452, 466, 494, 504
StartUpParameters property 336
State element 452, 465, 470
state information 152
static data 82, 286, 316
StaticDataRef element 317
status code 584
Status element
CancelJob operations 245
CancelReport operations 246
ExecuteQuery operations 268
ExecuteReport operations 271
ExecuteVolumeCommand operations 272
GetJobDetails operations 62, 305
GetNoticeJobDetails operations 308
GetSyncJobInfo operations 319
WaitForExecuteReport operations 405
status information 47, 432, 588
See also Status element; status messages
status messages
cancelled jobs and 64
channel notifications and 57
delete operations and 127

returning 158
setting polling interval for 455, 480
transaction operations and 190, 233
wait-time generated delays and 49
Status property 303, 307
StatusCode element 213
StatusCode entry 584, 588
StatusDescription entry 588
StatusErrorCode element 511
StatusErrorDescription element 511
Stop operations 557
Stop type definition 557
stop_consolidator.sh 578
StopArchive command 272
stopping
log consolidator application 576, 578, 579
report execution 246
RSSE applications 557
usage and error logging 593, 594
Stopping element 510
Stopping value 511
StopResponse type definition 557
Stream data type 513
Stream element 316
streaming 108, 190, 233, 513
StreamName element 286
streams 90, 235, 286
String data type 429, 482
String element 429, 505
String parameter 505
string tables 627
strings
access control lists and 542
creating job schedules and 469
encryption and 632
limitations for characters in 643
naming report files and 436
passwords and 516
Ping requests and 146
user names and 515
Structure data type 482
Structure element 429
style sheets 82, 286, 317
StyleSheetRef element 318
subdirectories 116
subfolders
assigning privileges to 116
creating archives and 599, 600
deleting 258
searching 116, 248, 302, 329, 345
setting archiving rules for 604
SubmissionTime element 466, 487, 504
submit job events 572
SubmitJob element 53, 54, 57, 60
SubmitJob operations
archiving and 605
defining 361
printing requests and 52, 54
running reports and 39, 52, 54, 207
sending notifications and 56, 57, 59
specifying resource groups in 74
SubmitJob type definition 362
SubmitJobResponse element 56
SubmitJobResponse type definition 368
SubscribedToChannelId element 519
SubscribedToChannelName element 519
SubscribedUserId element 417
SubscribedUserName element 417
subscribers 57, 374, 375
SubscribeToChannelsById element 401
SubscribeToChannelsByName element 400
Succeeded element 414
Succeeded message 64, 158
Succeeded value 138
success notices 339, 366, 516, 556, 561, 612
success notification template 61
SuccessNoticeExpiration element 516, 561
SuccessNoticeExpiration property 556
SUM function 409
Summary logging level 600
SupportCollation element 490
SupportColorMode element 491
SupportDuplex element 491
SupportedQueryFeatures data type 513
SupportedQueryFeatures element
GetInfoObject operations 300
GetJobDetails operations 303
GetNoticeJobDetails operations 307
SupportNumberOfCopies element 490
SupportOrientation element 490
SupportPageSize element 490
SupportPaperTray element 491
SupportResolution element 490
SupportScale element 490

SVGFlag property 201
Sync value 496
SyncFactoryProcesses element 289
synchronous job status attributes 271, 319
synchronous jobs
  cancelling 414
  creating 503
  getting information about 27, 289, 318
  getting list of 152
  monitoring requests for 156
  pending 485
  running 65, 496, 509
synchronous report states 36
synchronous reports
  cancelling 156, 157, 158, 246
  generating 73
  monitoring 156
  running 46, 50, 198
synchronous resource groups 67, 314, 497
synchronous silent installations 634
SyncJobQueueSize element 289
SyncJobQueueWait element 155, 289
SyncResourceGroupList element 314
SyncStoreTimeout element 155
system administrators 21, 101, 632
  See also administrators
system component ID 582, 583
system component log records 588
system database 633, 635, 641
system errors 574
system information 138, 151
system passwords 101, 368
system printers 27, 140, 320, 403, 488
system resources 564
SystemComponentId entry 582, 583
SystemDefaultVolume element 322
system-generated tokens 328
SystemLogin element 101
SystemLogin operations 43, 101, 156, 368
SystemLogin type definition 368
SystemLoginResponse element 101
SystemLoginResponse type definition 368
SystemName element 322
SystemPassword element 368
SystemPasswordEncryptLevel element 368
SystemRestart element 322
SystemType data type 513
SystemType element 511

T

Table data type 482
Table element 429
table of contents 85, 301, 322
table parameters 435, 483, 485, 494
tables 575, 579
TableValue element 485
Target element 131, 132, 247, 329
target volumes 21, 101, 136, 173, 223
  See also Encyclopedia volumes
targetNamespace attribute 7
TargetResourceGroup element 21, 73, 240, 449
TargetResourceGroup variable 175
TargetServer element 21, 152, 240, 448
TargetVolume element 21, 101, 136, 241, 448
TargetVolume variable 175, 224
tcpmon utility. *See* Apache AXIS TCPMonitor
template files 639
templates
  sending notifications and 61
templates (ACLs) 103, 294
templates (notifications) 61
temporary files 148, 265, 334
temporary objects 47, 267
temporary reports 47, 198, 289, 340, 503
testing
  iHub components 28
  page-level security 543
  report generation 87
  Windows installations 624
text 201, 625, 632, 643
text boxes 434, 483
text files 342, 350
text formats 420
text strings. *See* strings
text wrapping attributes 420
TextFormat element 420
third-party code libraries 163
third-party files 51, 312, 381
third-party reports
  attaching to e-mail 57
  generating 46
  managing 4

selecting output formats for 56
uploading 107, 190, 233
threads 567, 568
Time data type 428
Time element 428, 505
Time parameters 505
time stamps 121, 436, 599
time zones 121, 467
time-out intervals 157
TimeStamp element 121, 436, 619
TimeStamp property 120, 298
TimeZoneName element 467
TocNodeId element 323
TocRef element 302, 323
tokens 328
TotalCount element
GetChannelACL operations 102, 277
GetFileACL operations 102, 294
GetFileCreationACL operations 295
GetFolderItems operations 299
GetParameterPicklist operations 311
SelectChannels operations 344
SelectFiles operations 346
SelectGroups operations 348, 553
SelectJobs operations 351, 352
SelectJobSchedules operations 354
SelectRoles operations 358, 554
SelectUsers operations 359, 554
TotalCount parameter 90
Trace logging level 600
Trace mode 150, 334
transaction applications 187, 188, 230, 231
Transaction element 58, 134, 243
transaction logs 567, 592
Transaction objects 189, 232
Transaction operations
Administrate operations and 31, 133, 188
creating 369
UpdateJobSchedule operations and 58
Transaction requests 187, 231
Transaction tag 187, 231
Transaction type definition 369
TransactionOperation element 135, 231, 369
TransactionOperation objects 189, 232
TransactionOperation operations 369
TransactionOperation requests 135, 187, 231
TransactionOperation type definition 369
transactions 132, 133, 187, 369, 576
See also Transaction operations
transfer protocols 4, 14
transient files. *See* temporary files
transient reports. *See* temporary reports
TransientReportCacheSize element 289
TransientReportTimeout element 289
TranslatedRoleNames data type 559
TranslatedRoleNames element 326, 549
TranslatedRoleNames property 139, 325
TSV parameter 341, 522
type descriptors 166
Type element
DatabaseConnectionDefinition type 424
GetDatabaseConnectionParameters operations 282
ObjectIdentifier data type 479
ResourceGroup type 66, 67, 496
ServerResourceGroupSetting type 509
type element 421
TypeName data type 514
TypeName element 421, 514
typens namespace prefix 7
types. *See* data types
types element 6, 7–8
typical installation 631

U

UI_Version_2 element 513
UNCSV parameter 342, 522
undefined parameters 252
UndeleteJobSchedule element 372
UndeleteUser element 371
UndeleteUser operations 372
Unicode character sets 16, 93
Uniform Resource Locators. *See* URLs
Uniform Resource Names. *See* URNs
uninstall log files 636, 641
uninstall scripts (UNIX) 641
uninstalling
iHub System 635, 641
localization and online documentation files 636
universal hyperlinks. *See* hyperlinks
Universal Resource Identifiers. *See* URIs
UNIX messaging protocol 4

UNIX systems
See also Linux servers
configuring archive provider for 602
creating notifications for 61
creating silent installs for 638, 639
customizing installations for 638
MAPI encoding for 61
running log consolidator application and 575, 576, 578
running logging extensions for 564
running sample applications and 171
running silent installs for 640–641
running silent uninstalls for 641
UnsubscribeFromChannelsById element 401
UnsubscribeFromChannelsByName element 400
UNTSV parameter 342, 522
update operations
composite operations and 30, 124
file or folder properties and 33, 377
handling errors with 124
job information and 40
managing Encyclopedia items and 31
notification groups and 34
user information and 44, 128
update requests 98
UpdateChannel element 130, 243, 371
UpdateChannel operations 130, 373
UpdateChannel type definition 373
UpdateChannelOperation element 376
UpdateChannelOperation operations 374
UpdateChannelOperation type definition 374
UpdateChannelOperationGroup element 373
UpdateChannelOperationGroup operations 376
UpdateChannelOperationGroup type definition 376
UpdateDatabaseConnection operations 35, 376
UpdateDatabaseConnection type definition 376
UpdateDatabaseConnectionResponse type definition 376
UpdateFile element 112, 244, 372
UpdateFile operations
archiving and 605, 606, 607, 608
defining 112, 377
updating privileges and 114
UpdateFile type definition 377
UpdateFileOperation element 381, 383
UpdateFileOperation operations 378
UpdateFileOperationGroup element 378
UpdateFileOperationGroup operations 381
UpdateFileOperationGroup type definition 378, 381
UpdateFileType element 244, 371
UpdateFileType operations 381
UpdateFileType type definition 381
UpdateFileTypeOperation element 383
UpdateFileTypeOperation operations 382
UpdateFileTypeOperationGroup element 381
UpdateFileTypeOperationGroup operations 383
UpdateFileTypeOperationGroup type definition 382, 383
UpdateGroup element 243, 371
UpdateGroup operations 383
UpdateGroup type definition 383
UpdateGroupOperation element 386
UpdateGroupOperation operations 384
UpdateGroupOperation type definition 384
UpdateGroupOperationGroup element 384
UpdateGroupOperationGroup operations 385
UpdateGroupOperationGroup type definition 386
UpdateJobSchedule element 129, 244, 372
UpdateJobSchedule operations
archiving and 605, 608
changing properties and 129
defining 386
Factory service tasks and 53
running reports and 207
sending notifications and 56, 58, 60
UpdateJobSchedule type definition 386
UpdateJobScheduleOperation element 391
UpdateJobScheduleOperation operations 387
UpdateJobScheduleOperation type definition 387

UpdateJobScheduleOperationGroup element 386
UpdateJobScheduleOperationGroup operations 391
UpdateJobScheduleOperationGroup type definition 391
UpdateOpenSecurityCache element 244, 372
UpdateOpenSecurityCache operations 44, 391
UpdateOpenSecurityCache type definition 391
UpdateResourceGroup element 68
UpdateResourceGroup operations 68, 392
UpdateResourceGroup type definition 392
UpdateResourceGroupResponse type definition 392
UpdateRole element 98, 243, 371
UpdateRole operations 393
UpdateRole type definition 393
UpdateRoleOperation element 397
UpdateRoleOperation operations 394
UpdateRoleOperation type definition 394
UpdateRoleOperationGroup element 393
UpdateRoleOperationGroup operations 396
UpdateRoleOperationGroup type definition 396
updates 20
UpdateUser element 134, 243, 371
UpdateUser operations 397
UpdateUser type definition 397
UpdateUserOperation element 402
UpdateUserOperation operations 398
UpdateUserOperation type definition 398
UpdateUserOperationGroup element 397
UpdateUserOperationGroup operations 401
UpdateUserOperationGroup type definition 402
UpdateVolumeProperties element 244, 372
UpdateVolumeProperties operations 402, 606, 609
UpdateVolumeProperties type definition 402
UpdateVolumePropertiesOperation element 404
UpdateVolumePropertiesOperation operations 402
UpdateVolumePropertiesOperation type definition 402
UpdateVolumePropertiesOperationGroup element 402
UpdateVolumePropertiesOperationGroup operations 403
UpdateVolumePropertiesOperationGroup type definition 403
updating
  access control lists 116
  archiving rules 381, 605, 607–609
  channels 130, 373, 374, 376, 415
  file types 128, 129, 381, 382, 383
  files 112, 129, 377, 378, 381
  folders 112, 377, 378, 381
  multiple objects 124
  notification groups 129, 383, 384, 385
  Open Security cache 391
  privileges 112, 114, 390
  resource groups 361, 392
  roles 98, 393, 394, 396, 400
  schedules 129, 386, 387, 391
  user properties 397, 398, 401
upgrades 629, 638
upload applications 191, 234
UploadFile class 191, 234
UploadFile element 105
uploadFile method 191, 192
UploadFile objects 192, 235
UploadFile operations
  building applications and 192, 235
  copying file properties and 107
  defining 105, 106, 404
  managing files and folders and 34
  uploading third-party reports and 107
UploadFile requests 109, 110, 235
UploadFile responses 108
UploadFile type definition 404
UploadFileResponse type definition 404
uploading
  external files 105
  files 90, 105, 109, 190, 233, 404
  reports 106, 543
  third-party reports 107, 190, 233
URIs 16, 17, 193
URL property 217
URLs
  accessing web services and 10
  accessing WSDL documents and 11, 163

URLs *(continued)*
configuring log consolidator and 576
getting embedded 82
getting SOAP port 169, 170
logging in to Encyclopedia and 174
retrieving dynamic data and 285
setting file type icon 442
specifying media types and 16
text string limits for 643
URNs (Uniform Resource Names) 16
usage event IDs 580
Usage function 223
usage log consolidator 564, 574
usage log data 583, 588
usage log files 564, 567, 568, 588, 592
usage log settings 577
usage logging configurations 565
Usage Logging extension 564, 567, 592
usage logging functions 592
Usage Logging page 565
usage_log.csv 564, 568
UsageAndErrorConsolidator directory 575
usageanderrorconsolidator.jar 575, 576
UsageAndErrorLogging.rol 590
USAGELOG_FILE_EXT property 567
USAGELOG_FILE_NAME property 567
usagelogext.c 567
UseQuoteDelimiter element 507
UseQuoteDelimiter property 342, 350
user activity errors 573
user attributes 514, 517
User data type 514, 560, 645
User element
Authenticate operations 546
CreateUser operations 254, 612
GetUserProperties operations 550
Login requests 327
Login responses 328
user error messages 574
user IDs 264, 373, 458, 515
user names
adding to access control lists 542
connecting to external data sources and 548
consolidating logging information and 579
deleting 264
duplicating 254
getting 139
sending notifications and 459
setting 176, 515
specifying invalid 124
User objects 180, 226
See also users
user operations 42
user preferences 99, 180, 226, 514, 517
User property 120
user.acls 541, 542, 545
UserACLExternal element 556
User-Agent directive 16
UserAgent element 522
UserAgent parameter 342
UserAndProperties element 546
UserCondition data type 517
UserField data type 517
UserId element 324, 360, 488
UserName element
DoesUserExist operations 548
GetConnectionProperties operations 548
GetUserACL operations 550
GetUserLicenseOptions operations 324
GetUserPrinterOptions operations 324
Permission data type 487, 558, 620
SelectGroups operations 552
SelectRoles operations 553
SetConnectionProperties operations 360
UserName property 335
username variable 173, 223
UserNames element 277
UserPermissions element 415, 436
UserPermissions property 298
users
adding to notification groups 385, 388
assigning privileges 487, 619
authenticating 326, 546
checking for existence of 547
creating 125, 179, 226, 254, 543
defining access control lists for 542
defining available features for 328
defining groups of 251
defining login settings for 99
deleting 128, 263, 372
developing for external 530
getting access control lists for 541

getting ACL templates for 103, 294
getting job information for 61, 63
getting licensing options for 323, 519
getting list of 136
getting printer settings for 139, 142, 324
getting privileges for 102, 104
getting properties for 550
identifying 103
managing volume data and 30, 99
programming tasks for 42
removing from notifications 385, 388
searching 347, 358, 517, 518
sending notifications to 57, 58, 125, 338, 365
setting home folders for 125
setting licensing options for 474
setting notification options for 56, 59
setting output format options for 57, 59
setting passwords for 516
setting preferences for 99, 180, 226, 514, 517
setting properties for 556
updating properties for 397, 398, 401
updating roles for 395
viewing error messages for 574
Users element 359, 551, 554
UserSearch data type 518
UserSetting element 327, 546
useSOAPAction parameter 193
UTF-8 character set 16
utils package 177

V

ValidateRole element 327
ValidateRoles element 99, 100
validating security roles 99
ValidRoles element 328
Value element
Argument data type 410
ComponentType data type 423
FieldValue data type 435
FilterCriteria data type 443
LicenseOption data type 474
NameValuePair data type 477
ParameterValue data type 485
PropertyValue data type 494, 559
Value property 50
ValueFileType element 454
ValueFileVersionName element 454
ValueIsNullValue element 485
values
See also data
assigning to data components 423, 477
retrieving date 51
setting parameter 480, 484
setting property 493, 559
variables 639
version attributes 478, 508
Version element 436, 479, 555, 618, 630
version information 511, 630
version names 436, 478, 619
version numbers 54, 102, 105
version options 519
Version property 120, 298
VersionInfo element 629, 630
Versioning element 478
VersioningOption data type 519
VersionName element 436, 478, 619
VersionName property 120, 298
View element 495
view formats 299, 444
view parameters 313, 341, 484, 495, 520
See also ViewParameter element
View service
creating e-mail attachments and 56
enabling 512
getting formats for 87, 88
overview 76
pinging 28, 147, 148, 333, 334
viewer preferences 99, 524, 556
viewing
file properties 116
folder properties 116
Reportlets 279
reports 76, 77, 201, 520
search results 83
SOAP messages 177
specific report pages 77
WSDL schema definitions 11
Viewing element 512
viewing events 569, 571
viewing formats 87
See also formats

viewing options
  reports 367, 521
  search result sets 341
viewing parameters. *See* view parameters
viewing preferences. *See* viewer preferences
ViewMode element 77, 480
viewMode property 201
ViewOperation element 522
ViewParameter data type 520
ViewParameter element
  GetContent operations 278
  GetCustomFormat operations 88
  GetCustomFormatData operations 280
  GetDynamicData operations 284
  GetStaticData operations 316
  GetStyleSheet operations 317
  GetTOC operations 323
  SearchReport operations 341
  SelectPage operations 355
ViewParameterList element 313
ViewParameterValues element 349
ViewPreference element 516, 560
ViewPreference property 556
ViewProperties element 349
ViewResourceGroupList element 314
view-time parameters 443
Visibililty element 421
Visible attribute 631
visible privilege 488, 620
Vista computers. *See* Windows systems
Visual Basic development environments 216
volume administrators 30, 98
  See also administrators
Volume archive service provider property 603
Volume data type 523
Volume element 486, 496, 503, 555
volume failover errors 573
volume health monitoring errors 573
volume job purging errors 573
volume-level operations 31, 105
volume names 28, 321
volume online or offline errors 573
volume user activity errors 573
volume variable 173
VolumeName element 213, 272
volumename variable 223
VolumeNameList element 322
VolumeProperties element 326
VolumeProperties property 139, 325
volumes. *See* Encyclopedia volumes

W

wait intervals 48, 270
WaitForEvent element 305, 309, 368
WaitForExecuteReport element 49
WaitForExecuteReport operations 49, 405
WaitForExecuteReport type definition 405
WaitForExecuteReportResponse element 50
WaitForExecuteReportResponse type definition 405
WaitTime element 48, 268, 270
WaitTime requests 48
WaitTime responses 48
WARN logging level 531
warnings 572
Warnings element 250, 282, 376
web applications 4
  See also applications
web browsers 342, 383, 522
web icons 383, 442
web service attributes 5, 6
web service definitions 10–11
Web Service Description Language. *See* WSDL
web service interfaces 167, 221
web service namespace declaration 7
web service operations 5, 10, 212
web service sample application 206
web services
  accessing 10, 14
  Apache Axis clients and 162
  calling 14–15
  creating 10
  defined 4
  deploying 4, 210
  developing for 4, 5, 9
  enabling for RSSE applications 538
  event-based scheduling and 206, 209, 210
  integrating with iHub 5
  Microsoft .NET clients and 216
  naming 10

page-level security and 540, 541
polling 207
running remote 168, 221
running RSSE applications as 530, 532, 533
setting content types for 16
Weekly data type 525
weekly reports 54, 525
Weekly value 469
Welcome dialog box 630
WelcomeDlg element 630
well-formed messages 14
wildcard characters 98, 183, 228
Windows messaging protocol 4
Windows Registry. *See* Registry Editor
Windows systems
changing registry keys for 622
configuring archive provider for 602
creating notifications for 61
creating silent installs for 628–633
customizing installations for 622–624
localizing installations for 624–628
MAPI encoding for 61
running log consolidator application and 575, 576, 578
running logging extensions for 564
running sample applications and 170
running silent installs for 633–635
running silent uninstalls for 635, 636
testing installations for 624
WithLicenseOption element 519
WithoutDynamicPickList element 312
WithRightsToChannelId element 501
WithRightsToChannelName element 501
WithRoleId element 519
WithRoleName element 519
WithUserId element 447
WithUserName element 447
word wrapping 420
working directories 131
WorkingFolderId element
CopyFile operations 248
CreateFolder operations 251
DeleteFile operations 258
MoveFile operations 329
SelectFiles operations 345
UpdateFile operations 377
WorkingFolderName element
CopyFile operations 248
CreateFolder operations 251
DeleteFile operations 258
MoveFile operations 329
SelectFiles operations 345
UpdateFile operations 377
workspace directories 113, 479
Wrap element 420
write privilege 488, 620
WriteFile requests 334
WriteFile value 147
WSDL (defined) 5
WSDL documents
accessing 163
Apache Axis environments and 162
creating SDIs and 167
creating web service interfaces and 221
generating C# classes from 219
generating JavaBeans from 165
Microsoft .NET clients and 216
subclassing IDAPI classes and 14
WSDL files
accessing 11
binding definitions in 9
data type definitions in 7–8, 411
message definitions in 9
namespace declarations in 6
portType definitions in 9
scoping messages to 7
service definitions in 10–11
structure described 5
updating 217
WSDL interface 217
WSDL schemas
accessing 11
case sensitivity in 5
creating 5–11
defining data type arrays and 411
defining operations with 5
development environments for 9
displaying operation-specific 11
omitting responses and 9
overview 5
WSDL2Java code emitter 163
WSDL2Java tool 165, 166, 167
wsdlns namespace prefix 7

X

x-axis coordinates (charts) 285, 287
Xerces XML Parser 164
XML documents 17, 18, 80
XML element 450
XML elements
  binding definitions and 10
  case sensitivity for 5
  character limits for 643
  defining data types and 8
  HTTP headers and 16
  mapping to Java types 166
  multiple data sources and 18
  SOAP messaging and 5
XML files 628
XML formats 80, 521
XML language support 4
XML namespace 17, 18
XML objects 575
XML Parser 164
XML parsing error messages 574
XML reports. *See* XML documents
XML schemas 5, 7, 18, 411
  See also WSDL schemas
XMLDisplay parameter 342, 522
XP computers. *See* Windows systems
xsd namespace prefix 7
xsi namespace 18
xsi type attributes 193

Y

y-axis coordinates (charts) 285, 287