EVIDEN

Identity and Access Management

# DirX Identity

## Connectivity Administration Guide

Version 8.10.17, Edition November 2026



All product names quoted are trademarks or registered trademarks of the manufacturers concerned.

© 2026 Atos Group

All Rights Reserved

Distribution and reproduction not permitted without the consent of Atos Group.

# Table of Contents


Copyright ii
Preface 1
DirX Identity Documentation Set 2
Notation Conventions 4
1. Managing DirX Identity Connectivity 6
1.1. Managing Connected Directories 6
1.1.1. Managing Provisioning Workflows 6
1.2. Managing Passwords 7
1.3. Managing DirX Identity Servers 7
1.4. Managing the Connectivity System 7
1.5. About Java-based Configuration Objects 8
1.6. About Tcl-based Configuration Objects 10
2. Planning for DirX Identity Connectivity 12
2.1. Identify the Source Systems 12
2.2. Decide on Source System Workflows 12
2.3. Identify the Target Systems 12
2.4. Decide on Target System Workflows 12
2.5. Decide on Password Synchronization 13
2.6. Define the System Architecture 13
2.7. Define System Maintenance 13
3. Managing Connected Directories 14
3.1. Setting Up the Connected Directory Structure 14
3.2. Setting Up Connected Directory Authentication Information 14
3.3. Schema and Attribute Configuration Handling 15
3.3.1. Schema Handling for LDAP Directories 15
3.3.2. Schema Handling for Other Databases 15
3.3.3. Schema Handling for Fixed Directories and Databases 16
3.3.4. Schema Handling for Files 16
4. Managing Provisioning Workflows 17
4.1. Managing Java-based Provisioning Workflows 17
4.1.1. Java-based Workflow Runtime Operation 17
4.1.1.1. General Runtime Operation 17
4.1.1.2. Server Support for Runtime Operation 18
4.1.2. Target System Provisioning Workflow Operation 21
4.1.3. Error Handling and Retry Operation 22
4.1.4. Using Cluster Workflows 22
4.1.4.1. Setting Up Non-clustered and Clustered Workflows 23
4.1.4.2. Configuring Cluster Workflows 25
4.1.4.2.1. About Workflow Configuration Attributes 25

4.1.4.2.2. About Connector Configuration Attributes . . . 26
4.1.4.2.3. About Environment Configuration Attributes . . . 27
4.1.4.3. Scheduling Cluster Workflows . . . 27
4.1.5. Copying Java-based Provisioning Workflows . . . 27
4.1.5.1. Copying Java-based Workflows in the Global View . . . 27
4.1.5.2. Copying Java-based Workflows in the Expert View . . . 28
4.1.6. Starting Java-based Provisioning Workflows . . . 29
4.2. Managing Tcl-based Provisioning Workflows . . . 29
4.2.1. Runtime Operation . . . 29
4.2.1.1. Basic Runtime Operation . . . 30
4.2.1.2. Agent Controller Runtime Tasks . . . 31
4.2.1.3. Workflow and Activity Execution Status Values . . . 33
4.2.1.4. Workflow Execution Status . . . 33
4.2.1.5. Activity Execution Status . . . 34
4.2.1.6. Checkpoints . . . 35
4.2.1.7. File Paths and Areas . . . 36
4.2.2. Configuration Files and Tcl Scripts . . . 38
4.2.3. Object Naming . . . 39
4.2.3.1. Object Identifiers (common names) . . . 39
4.2.3.2. Display Names . . . 40
4.2.4. Linking Objects . . . 40
4.2.5. Understanding Notifications . . . 41
4.2.5.1. Change Notifications . . . 41
4.2.5.2. E-mail Notifications . . . 41
4.2.5.3. Notify if not OK . . . 43
4.2.5.4. Data Notification . . . 43
4.2.5.5. Creating Your Own Notifications . . . 44
4.2.6. Workflow Design Rules . . . 44
4.2.6.1. Rules for Algorithms . . . 45
4.2.6.1.1. Single Master System Design . . . 45
4.2.6.1.2. Multi Master System Design . . . 45
4.2.6.2. Rules for Schedules . . . 45
4.2.6.2.1. Starting the Same Workflow Twice . . . 46
4.2.6.2.2. Starting Workflows after the Server Starts . . . 46
4.2.6.2.3. Basic Rules for Central Configuration Object Parameters . . . 46
4.2.6.2.4. Rule 1: Polling time . . . 46
4.2.6.2.5. Rule 2: Time Interval . . . 46
4.2.6.2.6. Rule 3: Schedule Sync Interval . . . 46
4.2.6.2.7. Basic Rules for Schedule Object Parameters . . . 47
4.2.6.2.8. Rule 4: Time Interval . . . 47
4.2.6.2.9. Rule 5: Deviation . . . 47
4.2.6.2.10. Combined Rules . . . 47

4.2.6.2.11. Rule 6: Non overlapping start times for workflows . . . 48
4.2.6.2.12. Rule 7: Non overlapping execution of workflows . . . 49
4.2.6.3. Rules for Backup and Restore . . . 49
4.2.7. Copying Tcl-based Provisioning Workflows . . . 50
4.2.7.1. Copying Workflows in the Global View . . . 50
4.2.7.2. Copying Workflows in the Expert View . . . 52
4.2.8. Starting Tcl-based Provisioning Workflows . . . 52
5. Managing Passwords . . . 54
5.1. Understanding Password Management . . . 54
5.2. Password Changes from Windows Domains . . . 56
5.2.1. Password Changes via the Windows Password Listener . . . 56
5.2.2. About the Windows Password Listener . . . 57
5.3. Password Changes from Web Applications . . . 58
5.4. Configuring the Password Synchronization Workflows . . . 59
5.5. Understanding Password Synchronization Workflow Operation . . . 61
5.6. About the Password Change Algorithms . . . 62
6. Managing DirX Identity Servers . . . 65
6.1. Managing the Message Broker . . . 65
6.1.1. Planning the Message Broker Deployment . . . 66
6.1.2. About the Message Broker Components . . . 66
6.1.3. Starting the Message Broker . . . 67
6.1.4. Configuring the Message Broker . . . 67
6.1.5. Monitoring the Message Broker . . . 67
6.1.6. Message Broker Logging . . . 68
6.1.7. Message Broker Instance Naming . . . 68
6.1.8. JMX Access to the Message Broker . . . 69
6.1.9. Understanding the Java Messaging Service . . . 69
6.1.10. Using Messages in DirX Identity . . . 70
6.1.10.1. Queues of the Java-based Server . . . 70
6.1.10.2. Topics of the Java-based Server . . . 72
6.1.10.3. Topics of the C++-based Server . . . 73
6.1.10.4. Topics of the Password Listener . . . 73
6.1.10.5. About Message Sizes . . . 74
6.1.10.5.1. Password Messages . . . 74
6.1.10.5.2. Real-time Events . . . 74
6.1.10.5.3. Command and Status Messages . . . 74
6.1.10.5.4. File Transfer Messages . . . 75
6.1.10.6. Messaging Subscriptions . . . 75
6.1.10.6.1. Java-based Server . . . 76
6.1.10.6.2. C++-based Server . . . 76
6.1.10.6.3. Meta Controller . . . 76
6.1.10.6.4. Manager . . . 77

6.2. Managing the Java-based Server . . . 77
6.2.1. Server Components . . . 78
6.2.1.1. Administration Components . . . 78
6.2.1.2. Adapters . . . 79
6.2.1.3. Request Workflows . . . 79
6.2.1.4. Workflow Engine and Connectors . . . 80
6.2.1.5. Handlers . . . 80
6.2.2. Server Processes . . . 81
6.2.2.1. Starting the Processes . . . 81
6.2.2.2. Configuring the Processes . . . 81
6.2.2.2.1. Java-based Server INI File Parameters . . . 81
6.2.2.2.2. Java-based Server Startup Script on Linux . . . 82
6.2.2.2.3. Java-based Server Password File Parameters . . . 82
6.2.2.3. Starting the Java-based Server in Suspended Mode . . . 84
6.2.3. Recovery . . . 85
6.2.4. Auditing . . . 85
6.2.5. Statistics . . . 86
6.2.6. Logging . . . 86
6.2.7. Naming Schemes . . . 87
6.2.7.1. Java-based Server Object Naming . . . 87
6.2.7.2. Service Naming . . . 87
6.2.7.3. File Folder Naming . . . 88
6.2.8. Resource Families . . . 88
6.2.8.1. Understanding the Pre-Configured Resource Families . . . 88
6.2.9. JMX Access to the Java-based Server . . . 89
6.3. Managing the C++-based Server . . . 90
6.3.1. Server Components . . . 90
6.3.1.1. Starting Up the Server Components . . . 90
6.3.1.2. Configuring the C++-based Server . . . 91
6.3.1.3. Changing the Service Login Account (Windows) . . . 92
6.3.1.4. Server Password Handling . . . 93
6.4. Distributed Deployments and Scalability . . . 93
6.4.1. All on One Machine . . . 94
6.4.2. Distributing Java-Based Servers . . . 94
6.4.2.1. Distribution Criteria . . . 95
6.4.2.2. Configuring One Java-based Server per Domain . . . 96
6.4.2.3. Configuring Multiple Java Servers per Domain . . . 96
6.4.3. Separating Traffic for Selected Connected Systems . . . 97
6.4.4. Distributing C++-based Server Components . . . 100
6.4.4.1. All Items on Central Server . . . 100
6.4.4.2. Target Activity Is Distributed . . . 102
6.4.4.3. All Items on Target Server . . . 103

6.4.4.4. Reduce File Handling in Status Area . . . 104
6.4.4.5. Setting Up a Shared File System for Distribution . . . 105
6.4.5. Distributing Message Broker Instances . . . 105
6.5. High Availability and Recovery . . . 105
6.6. Diagnostic Information . . . 106
6.7. Managing Daylight Savings Time . . . 107
6.8. Connector Frameworks . . . 107
6.8.1. Identity Connector Framework for Java . . . 108
6.8.2. Identity Connector Framework for C/C++ . . . 109
7. Managing the Connectivity System . . . 111
7.1. Managing Administrative Accounts . . . 111
7.1.1. Changing the DomainAdmin Password (Provisioning) . . . 112
7.1.2. Changing the SystemAdmin Password (Provisioning) . . . 113
7.1.3. Changing the Connectivity server_admin Password . . . 114
7.1.4. Changing the Connectivity admin Password . . . 114
7.1.5. Changing the Embedded Tomcat admin Password . . . 115
7.1.6. Changing the ActiveMQ admin Password . . . 115
7.2. Managing Connectivity Security . . . 116
7.2.1. Securing DirX Identity Data . . . 116
7.2.1.1. Securing Control Data . . . 116
7.2.1.2. Securing Password Changes . . . 117
7.2.1.3. Securing Configuration Data . . . 117
7.2.1.4. Securing Administrative Passwords . . . 117
7.2.1.5. Securing Data to be Synchronized . . . 117
7.2.1.6. Securing Auditing Information . . . 118
7.2.1.7. About the Crypto Algorithms used by DirX Identity . . . 118
7.2.2. Managing Data Encryption . . . 118
7.2.2.1. How DirX Identity Data Encryption Works . . . 118
7.2.2.2. Handling Specific Encryption Requirements . . . 120
7.2.2.3. Setting up Data Encryption . . . 121
7.2.2.3.1. Extending/Creating Workflows for Encrypted Synchronization . . . 122
7.2.2.3.2. Initially Encrypting Administrative Passwords . . . 123
7.2.2.3.3. Initially Encrypting Attributes . . . 123
7.2.3. Setting up Audit Signature . . . 124
7.2.4. Generating a Personal Security Environment (PSE) . . . 124
7.2.5. Managing Anonymously Readable Attributes . . . 126
7.2.6. Managing Keys . . . 127
7.2.6.1. Managing Keys for Data Encryption . . . 127
7.2.6.2. Using the Key Migration Tool . . . 128
7.2.6.3. Managing Keys for Audit Signature . . . 129
7.3. Establishing Secure Connections with SSL . . . 129
7.3.1. Securing Connectivity Database Connections with SSL . . . 130

7.3.1.1. Setting up the LDAP Server . . . 131
7.3.1.1.1. Managing Keys . . . 131
7.3.1.1.2. Preparing the LDAP Server for SSL . . . 131
7.3.1.1.3. Important Locations . . . 131
7.3.1.2. Setting up DirX Identity with Initial Configuration . . . 131
7.3.1.2.1. Managing Keys . . . 131
7.3.1.2.2. Preparing the Certificate Stores . . . 132
7.3.1.2.3. Using Initial Configuration to Set up DirX Identity . . . 133
7.3.1.2.4. Important Locations . . . 133
7.3.1.3. Setting up Identity Manager . . . 133
7.3.1.3.1. Managing Keys . . . 133
7.3.1.3.2. Setting up Manager for SSL (server-side SSL) . . . 134
7.3.1.3.3. Important Locations . . . 134
7.3.1.4. Setting up Web Center . . . 134
7.3.1.4.1. Managing Keys . . . 134
7.3.1.4.2. Important Locations . . . 135
7.3.1.5. Setting up the Java-based Server . . . 135
7.3.1.5.1. Managing Keys . . . 135
7.3.1.5.2. Setting up the Java-based Server for SSL . . . 136
7.3.1.5.3. Important Locations . . . 136
7.3.1.6. Setting up the C++-based Server . . . 136
7.3.1.6.1. Managing Keys . . . 136
7.3.1.6.2. Setting up the C++-based Server for SSL . . . 136
7.3.1.6.3. Important Locations . . . 137
7.3.1.7. Setting up the Java-based Configuration Wizard . . . 137
7.3.1.7.1. Managing Keys . . . 137
7.3.1.7.2. Setting up the Java-based Configuration Wizard for SSL . . . 137
7.3.1.7.3. Important Locations . . . 137
7.3.2. Securing Provisioning Database Connections with SSL . . . 138
7.3.2.1. Setting up the LDAP Server . . . 138
7.3.2.2. Setting up DirX Identity with Initial Configuration . . . 138
7.3.2.3. Setting up Identity Manager . . . 138
7.3.2.4. Setting up Web Center . . . 138
7.3.2.4.1. Managing Keys . . . 138
7.3.2.4.2. Setting up Web Center for SSL (server-side SSL) . . . 139
7.3.2.4.3. Setting up Web Center for Password Management for SSL (server-side SSL) . . . 139
7.3.2.5. Setting up the Java-based Server . . . 139
7.3.2.6. Setting up the Web Services . . . 139
7.3.2.6.1. Managing Keys . . . 139
7.3.2.6.2. Setting up the Web Services for SSL . . . 140
7.3.2.6.3. Important Locations . . . 140

7.3.2.7. Setting up the Meta Controller . . . 140
7.3.2.7.1. Important Locations . . . 140
7.3.2.8. Setting up the Java-based Configuration Wizard . . . 140
7.3.3. Securing Identity Server Connections with SSL . . . 141
7.3.3.1. Setting up the X.509 Certificates . . . 141
7.3.3.1.1. Configuring the Certificate Generation Scripts . . . 142
7.3.3.1.2. Setting up a Certificate Authority . . . 142
7.3.3.1.3. Setting up the Host Server Key . . . 143
7.3.3.1.4. Signing the Server Certificate Request . . . 143
7.3.3.1.5. Importing the Server's Certificate into the Shared Key Store . . . 143
7.3.3.1.6. Converting the Server Key to PEM . . . 144
7.3.3.1.7. Setting up the Client Key . . . 144
7.3.3.1.8. Signing the Client Certificate Request . . . 144
7.3.3.1.9. Importing the Client Certificate into the Shared Key Store . . . 144
7.3.3.1.10. Converting the Client Key to PEM . . . 145
7.3.3.1.11. Importing the Certificate into the Java for DirX Identity . . . 145
7.3.3.2. Securing the Identity Services . . . 146
7.3.3.2.1. Securing DirX Identity Manager with SSL . . . 146
7.3.3.2.2. Securing the Java-based Server with SSL . . . 146
7.3.3.2.3. Securing the C++-based Server with SSL . . . 146
7.3.3.2.4. Securing the Message Broker and Web Console with SSL . . . 147
7.3.3.2.5. Securing Web Admin and Server Admin with SSL . . . 148
7.3.3.2.6. Securing Web Center with SSL . . . 148
7.3.3.2.7. Securing the Supervisor with SSL . . . 150
7.3.3.2.8. Securing Provisioning Web Services with SSL . . . 150
7.3.3.2.9. Securing the Business User Interface with SSL . . . 151
7.3.3.2.10. Securing the Windows Password Listener with SSL . . . 151
7.3.3.2.11. Securing the Meta Controller with SSL . . . 151
7.3.3.2.12. Securing JMX Clients with SSL . . . 152
7.3.3.2.13. Securing UNIX Scripts with SSL . . . 152
7.3.4. Securing Connections between Web Services Clients and Web Services with SSL . . . 152
7.3.4.1. Managing Keys for Server-Side SSL . . . 152
7.3.4.2. Managing Keys for Client-Side SSL . . . 153
7.3.4.3. Setting up SSL for Web Services Clients for Server-Side SSL . . . 153
7.3.4.4. Setting up SSL for Web Services Clients for Client-Side SSL . . . 153
7.3.4.5. Important Locations . . . 153
7.3.5. Securing Connections to Tomcat Web Applications with SSL . . . 153
7.4. Understanding File-Handling Mechanisms . . . 154
7.4.1. Path Handling . . . 155
7.4.2. File Preservation Mechanisms . . . 155
7.4.3. File Transfer Mechanisms . . . 156

7.4.3.1. Local Machine File Handling . . . 156
7.4.3.2. Distributed Machines with Shared File Systems . . . 156
7.4.3.3. Distributed Machines with No Shared File System Set Up . . . 157
Appendix A: Context-Sensitive Help . . . 158
A.1. General . . . 158
A.1.1. Content . . . 158
A.1.2. Data File . . . 158
A.1.3. Files . . . 160
A.1.4. File Item . . . 161
A.1.5. Query Folder . . . 162
A.1.6. Specific Attributes . . . 164
A.1.7. Tcl Content . . . 165
A.1.8. XML Content . . . 165
A.1.9. XML File . . . 166
A.1.10. Object Descriptions . . . 166
A.1.11. Wizards . . . 167
A.2. Agents . . . 167
A.2.1. Agent . . . 167
A.2.2. Agents . . . 169
A.3. Collections . . . 169
A.3.1. Collection . . . 169
A.4. Collections . . . 170
A.4.1. Collection Rule . . . 171
A.4.2. Collection Rules . . . 174
A.5. (Central) Configuration . . . 175
A.5.1. Agent Types . . . 179
A.5.1.1. Agent Type . . . 179
A.5.2. Connected Directory Types . . . 180
A.5.2.1. Connected Directory Type . . . 180
A.5.3. Connector Types . . . 181
A.5.3.1. Connector Type . . . 182
A.5.4. DirX Identity Servers . . . 183
A.5.4.1. Java-based Server . . . 183
A.5.4.1.1. Adaptor - General . . . 183
A.5.4.1.2. Adaptor - Limits . . . 184
A.5.4.1.3. Adaptor - Configuration . . . 185
A.5.4.1.4. Java-Based Server - General . . . 186
A.5.4.1.5. Java-based Server - Domain . . . 187
A.5.4.1.6. Java-based Server - Connectors . . . 188
A.5.4.1.7. Java-based Server - Repository . . . 188
A.5.4.1.8. Java-based Server - Limits . . . 190
A.5.4.1.9. Java-based Server - Resource Families . . . 192

A.5.4.1.10. Java-based Server - HA (High Availability) . . . 193
A.5.4.1.11. Java-based Server - Status and Auditing . . . 193
A.5.4.1.12. Java-based Server - Configuration . . . 196
A.5.4.1.13. Domain for Identity Servers . . . 196
A.5.4.1.14. Manage Servers - Adaptors . . . 197
A.5.4.1.15. Manage Servers - Request Workflow Timeout Check . . . 197
A.5.4.1.16. Manage Servers - Supervision . . . 197
A.5.4.1.17. Manage Servers - Schedule . . . 198
A.5.4.2. C++-based Server . . . 198
A.5.4.2.1. C++-based Server - General . . . 198
A.5.4.2.2. C++-based Server - SOAP Listener . . . 199
A.5.4.2.3. C++-based Server - SPML Receiver . . . 200
A.5.4.2.4. C++-based Server - Configuration . . . 201
A.5.4.2.5. C++-based Server - Paths . . . 202
A.5.4.2.6. C++-based Server - Agents . . . 203
A.5.4.2.7. C++-based Server - Agent Server State . . . 204
A.5.4.2.8. C++-based Server - JMX Access . . . 204
A.5.4.2.9. C++-based Server - Connector . . . 205
A.5.4.2.10. Get Server State . . . 206
A.5.5. GUI . . . 207
A.5.5.1. Extensions . . . 208
A.5.5.2. Report . . . 208
A.5.5.3. Report Definition . . . 209
A.5.5.4. Report Properties . . . 210
A.5.5.5. Reports . . . 210
A.5.6. JavaScripts . . . 210
A.5.6.1. JavaScript Content . . . 211
A.5.6.2. JavaScript File . . . 211
A.5.7. Messaging Services . . . 212
A.5.7.1. Messaging Service - Failover Transport Options . . . 212
A.5.7.2. Message Broker . . . 213
A.5.7.3. Message Broker - Transport Options . . . 213
A.5.7.4. Status Tracker (Topic) . . . 214
A.5.8. Resource Families . . . 214
A.5.8.1. Resource Family . . . 215
A.5.9. Services . . . 215
A.5.9.1. Service . . . 215
A.5.9.2. Services . . . 217
A.5.10. Standard Files . . . 217
A.5.11. Supervisors . . . 218
A.5.11.1. Supervisor . . . 218
A.5.11.2. Supervisor - Configuration . . . 219

A.5.11.3. Java Supervisor Mail . . . 220
A.5.11.4. Supervisor - Server . . . 220
A.5.11.5. Supervisors . . . 221
A.5.12. Systems . . . 221
A.5.12.1. System . . . 221
A.5.12.2. Systems . . . 223
A.5.13. Tcl . . . 223
A.5.13.1. Mapping Function . . . 223
A.5.13.2. Mapping Functions . . . 224
A.5.13.3. Tcl Folder . . . 225
A.5.14. Topics . . . 225
A.5.14.1. Topic . . . 226
A.6. Connected Directories . . . 226
A.6.1. Attribute Configuration - Details . . . 226
A.6.2. Attribute Configuration - General . . . 230
A.6.3. Attribute Configuration Template . . . 230
A.6.4. Bind Profile . . . 231
A.6.5. Bind Profile Container . . . 233
A.6.6. Bind Profiles . . . 233
A.6.7. Channels . . . 234
A.6.8. Connected Directories . . . 234
A.6.9. Connected Directory . . . 234
A.6.9.1. Configuring the Viewer Command . . . 236
A.6.10. Operational Attributes . . . 237
A.6.11. Provisioning Attributes . . . 239
A.6.12. SAP ECC UM Parameters . . . 242
A.6.13. Proxy Server . . . 243
A.6.14. Schema - Object Classes and Attribute Types . . . 243
A.6.15. Schema - General Properties . . . 245
A.6.16. Specific . . . 245
A.6.16.1. JDBC - Configuration . . . 245
A.6.16.2. GoogleApps - Google API . . . 245
A.6.16.3. Office 365 - Graph API . . . 246
A.6.16.4. Salesforce - Salesforce . . . 247
A.6.16.5. OICF - OpenICF Connector Server . . . 247
A.7. Jobs . . . 248
A.7.1. Authentication . . . 248
A.7.2. Attribute Mapping . . . 249
A.7.3. Control Scripts . . . 249
A.7.4. Default Values . . . 250
A.7.5. Delta Handling . . . 250
A.7.6. Tcl-based Event Manager - Operation . . . 251

A.7.7. Tcl-based Event Manager - Tracing . . . 254
A.7.8. Tcl-based Event Manager - Workflows . . . 256
A.7.9. HDMS Parameters . . . 256
A.7.10. INI Content . . . 257
A.7.11. INI File . . . 257
A.7.12. Job . . . 258
A.7.13. Jobs . . . 260
A.7.14. Mapping Item . . . 260
A.7.15. Mapping Script . . . 261
A.7.16. Mapping Scripts . . . 262
A.7.17. Notification (Object) . . . 262
A.7.17.1. Notification Tab . . . 263
A.7.17.2. Content Tab . . . 263
A.7.18. Notification (Tab) . . . 264
A.7.19. Notifications . . . 265
A.7.20. Operation . . . 265
A.7.21. Other Operation Parameters . . . 267
A.7.22. Other Scripts . . . 269
A.7.23. Profile Scripts . . . 270
A.7.24. Tcl Script . . . 270
A.7.25. Tcl Scripts . . . 271
A.7.26. Tracing . . . 272
A.8. Monitoring . . . 276
A.8.1. Activity Status Data . . . 276
A.8.2. Monitor Folder . . . 277
A.8.3. Process Table . . . 278
A.8.4. Process Table Server . . . 278
A.8.5. Workflow Status - Data . . . 278
A.8.6. Workflow Status - Structure . . . 280
A.8.7. Workflow Status - Statistics . . . 280
A.8.8. Activity Status - Config . . . 282
A.8.9. Status Data . . . 282
A.8.10. Activity Status - Statistics . . . 283
A.8.11. Activity Status Trace . . . 284
A.9. Password Change . . . 285
A.9.1. Password Change Event Manager - Workflow . . . 285
A.9.2. Password Change Event Manager - Activity . . . 286
A.9.3. Password Change Application - Workflow . . . 287
A.9.4. Password Change Application - Activity . . . 288
A.9.5. Error Activity . . . 289
A.9.6. Error Notification . . . 289
A.10. Java-based Workflows . . . 290

A.10.1. Combined Java-based Workflow . . . 290
A.10.1.1. The Workflow Table . . . 291
A.10.1.2. The Toolbar . . . 291
A.10.2. Java-based Workflow . . . 291
A.10.3. Real-time Activity . . . 294
A.10.3.1. General . . . 294
A.10.3.2. Controller . . . 295
A.10.3.3. Notification (optional) . . . 296
A.10.4. Real-time Port . . . 296
A.10.5. Real-time Filter . . . 303
A.10.6. Real-time Channel . . . 304
A.10.6.1. General . . . 305
A.10.6.2. Import . . . 306
A.10.6.3. Export . . . 306
A.10.6.4. Delta . . . 309
A.10.6.5. Mapping . . . 311
A.10.6.6. Operational Mapping . . . 312
A.10.6.7. Join . . . 312
A.10.6.8. Primary Channel . . . 312
A.10.7. Content Tabs . . . 313
A.10.8. Real-time Java Mapping . . . 314
A.10.9. Java Mapping Editor . . . 315
A.10.9.1. The Mapping Table . . . 315
A.10.9.2. The Tool Bar . . . 317
A.10.10. Simple Expression Mapping . . . 319
A.10.11. Joining . . . 320
A.10.12. Service-Specific Pages . . . 324
A.10.12.1. Policy - Job Parameters . . . 324
A.10.12.2. Policy - Rule Search Parameters . . . 325
A.10.12.3. Policy - Object Search Parameters . . . 326
A.10.12.4. Service - Job Parameters . . . 327
A.10.12.5. Service - Import Properties . . . 328
A.10.12.6. Service - Requested Attributes . . . 328
A.10.12.7. Service - Limits . . . 329
A.10.13. Transport Workflows . . . 329
A.10.13.1. Transport - Connection . . . 329
A.10.13.2. Transport - Export . . . 330
A.10.13.3. Transport - Delete . . . 330
A.10.13.4. Transport - Import . . . 331
A.10.13.5. Transport - Domain Mapping . . . 332
A.10.13.6. Transport - Attribute Configuration New . . . 333
A.10.13.7. Transport - Attribute Configuration Old . . . 333

A.10.13.8. Transport - Attribute Mapping Choice . . . 336
A.10.14. Workflow-Specific Pages . . . 339
A.10.14.1. Campaign Generator Workflows . . . 339
A.10.14.1.1. Campaign Generator - Initiator . . . 339
A.10.14.1.2. Campaign Generator - Roles . . . 340
A.10.14.1.3. Campaign Generator - Permissions . . . 341
A.10.14.1.4. Campaign Generator - Groups . . . 343
A.10.14.2. Consistency Check Workflows . . . 345
A.10.14.2.1. User Resolution Workflow . . . 345
A.10.14.2.2. Mark Affected Users Workflow . . . 346
A.10.14.2.3. Consistency Check Workflow . . . 347
A.10.14.3. Event-based Maintenance Workflows - Event Attributes . . . 349
A.10.14.3.1. Event-based User Resolution Workflow . . . 349
A.10.14.3.2. Event-based Persona Resolution Workflow . . . 351
A.10.14.3.3. Event-Based Functional User Resolution Workflow . . . 352
A.10.14.3.4. Maintenance Workflows for Business Objects . . . 353
A.10.14.3.5. Maintenance Workflows for Accounts . . . 353
A.10.14.3.6. Generic Maintenance Workflows . . . 354
A.10.14.4. Event-based Maintenance - User Hook . . . 354
A.10.14.5. Joint Backup - Parameters . . . 355
A.10.14.6. Joint Backup - Post Operation . . . 355
A.11. Schedules . . . 356
A.11.1. Schedule . . . 357
A.11.2. Ranges . . . 359
A.11.3. Target System Cluster . . . 360
A.12. Scenarios . . . 360
A.12.1. Workflow Line . . . 360
A.12.2. Connected Directory Icon . . . 361
A.12.3. Scenario . . . 361
A.12.4. Scenarios . . . 363
A.13. Tcl-based Workflows . . . 363
A.13.1. Tcl-based Activities . . . 363
A.13.2. Tcl-based Activity . . . 364
A.13.3. Tcl-based Channel . . . 365
A.13.4. Data Flow Viewer . . . 367
A.13.5. Data Flow Editor . . . 370
A.13.6. Entry Handling . . . 371
A.13.7. Export Properties . . . 374
A.13.8. Import Properties . . . 381
A.13.9. Input/Output Channels . . . 386
A.13.10. Import to Identity Store Properties . . . 387
A.13.10.1. Standard Scripts . . . 387

A.13.10.1.1. Join Expressions and Filtering . . . 389
A.13.10.2. Previously Used Scripts . . . 390
A.13.11. Join to Identity Store Properties . . . 392
A.13.12. Other Properties of a Channel . . . 393
A.13.13. Register User . . . 394
A.13.14. Selected Attributes . . . 396
A.13.15. Superior Info . . . 396
A.13.16. Tcl-based Workflow . . . 397
A.13.17. Workflow - Operation . . . 400
A.13.18. Workflow - General Properties . . . 401
A.13.19. Workflows . . . 401
Appendix B: Mapping Functions . . . 403
B.1. addAttributes . . . 403
B.2. deleteAttributes . . . 403
B.3. hdmsCmd2dmsid . . . 403
B.4. hdmsdata2dn . . . 404
B.5. hdmsData2telno . . . 404
B.6. ifEqual . . . 404
B.7. ifNotEqual . . . 405
B.8. lADSpathCreate . . . 405
B.9. lBaseDNreplace . . . 406
B.10. lBool2Integer . . . 406
B.11. lDate2GMT . . . 406
B.12. lDNcreate . . . 407
B.13. lDNsplit . . . 408
B.14. lInteger2Bool . . . 409
B.15. listFirst . . . 409
B.16. listLast . . . 410
B.17. lListAppend . . . 410
B.18. lListFirst . . . 410
B.19. lListLast . . . 410
B.20. lListNth . . . 411
B.21. lListRest . . . 411
B.22. lPA2String . . . 411
B.23. lString2PA . . . 411
B.24. lStringAppend . . . 412
B.25. lStringCompose . . . 412
B.26. lStringConvertChars . . . 412
B.27. lStringEncrypt . . . 414
B.28. lStringEscape . . . 414
B.29. lStringEscapeLDIF . . . 415
B.30. lStringEscapeVar . . . 416

B.31. lStringModify . . . 416
B.32. lStringPrefix . . . 417
B.33. lStringRange . . . 417
B.34. lStringTrim . . . 417
B.35. lStringTrimLeft . . . 418
B.36. lStringTrimRight . . . 418
B.37. lStringUnescape . . . 418
B.38. lStringUnescapeVar . . . 419
B.39. lWordCapitalize . . . 419
B.40. lWordFirst . . . 419
B.41. lWordLast . . . 420
B.42. lWordNth . . . 420
B.43. RDNescape . . . 421
B.44. RDNunescape . . . 421
B.45. replaceAttributes . . . 421
B.46. StringAppend . . . 421
B.47. StringModify . . . 422
Legal Remarks . . . 424

# Preface

This manual describes the concepts and all details to administrate the DirX Identity Connectivity part.It consists of the following sections:

- Chapter 1 provides an overview about the tasks for managing DirX Identity Connectivity.
- Chapter 2 makes up a planning checklist of the issues that you must take into account when deciding how to deploy DirX Identity Connectivity in your environment.
- Chapter 3 describes how to manage connected directories.
- Chapter 4 describes how to manage provisioning workflows.
- Chapter 5 describes how to manage passwords.
- Chapter 6 describes how to manage DirX Identity servers.
- Chapter 7 describes how to manage the Connectivity system.
- Appendix A presents the context-sensitive help topics that are provided with DirX Identity (Connectivity).
- Appendix B provides information about mapping functions.

# DirX Identity Documentation Set

**Version 8.10.17 | Build 2130 | Date 2026-11-20**

The DirX Identity document set consists of the following manuals:

- *DirX Identity Introduction*. Use this book to obtain a description of DirX Identity architecture and components.
- *DirX Identity Release Notes*. Use this book to understand the features and limitations of the current release. This document is shipped with the DirX Identity installation as the file **release-notes.pdf**.
- *DirX Identity History of Changes*. Use this book to understand the features of previous releases. This document is shipped with the DirX Identity installation as the file **history-of-changes.pdf**.
- *DirX Identity Tutorial*. Use this book to get familiar quickly with your DirX Identity installation.
- *DirX Identity Provisioning Administration Guide*. Use this book to obtain a description of DirX Identity provisioning architecture and components and to understand the basic tasks of DirX Identity provisioning administration using DirX Identity Manager.
- *DirX Identity Connectivity Administration Guide*. Use this book to obtain a description of DirX Identity connectivity architecture and components and to understand the basic tasks of DirX Identity connectivity administration using DirX Identity Manager.
- *DirX Identity User Interfaces Guide*. Use this book to obtain a description of the user interfaces provided with DirX Identity.
- *DirX Identity Application Development Guide*. Use this book to obtain information how to extend DirX Identity and to use the default applications.
- *DirX Identity Customization Guide*. Use this book to customize your DirX Identity environment.
- *DirX Identity Integration Framework*. Use this book to understand the DirX Identity framework and to obtain a description how to extend DirX Identity.
- *DirX Identity Web Center Reference*. Use this book to obtain reference information about the DirX Identity Web Center.
- *DirX Identity Web Center Customization Guide*. Use this book to obtain information how to customize the DirX Identity Web Center.
- *DirX Identity Meta Controller Reference*. Use this book to obtain reference information about the DirX Identity meta controller and its associated command-line programs and files.
- *DirX Identity Connectivity Reference*. Use this book to obtain reference information about the DirX Identity agent programs, scripts, and files.
- *DirX Identity Troubleshooting Guide*. Use this book to track down and solve problems in your DirX Identity installation.
- *DirX Identity Installation Guide*. Use this book to install DirX Identity.

- *DirX Identity Migration Guide*. Use this book to migrate from previous versions.

# Notation Conventions

**Boldface type**
In command syntax, bold words and characters represent commands or keywords that must be entered exactly as shown.

In examples, bold words and characters represent user input.

*Italic type*
In command syntax, italic words and characters represent placeholders for information that you must supply.

[ ]
In command syntax, square braces enclose optional items.

{ }
In command syntax, braces enclose a list from which you must choose one item.

In Tcl syntax, you must actually type in the braces, which will appear in boldface type.

|
In command syntax, the vertical bar separates items in a list of choices.

...
In command syntax, ellipses indicate that the previous item can be repeated.

*userID_home_directory*
The exact name of the home directory. The default home directory is the home directory of the specified UNIX user, who is logged in on UNIX systems. In this manual, the home pathname is represented by the notation *userID_home_directory*.

*install_path*
The exact name of the root of the directory where DirX Identity programs and files are installed. The default installation directory is *userID_home_directory*/**DirX Identity** on UNIX systems and **C:\Program Files\DirX\Identity** on Windows systems. During installation the installation directory can be specified. In this manual, the installation-specific portion of pathnames is represented by the notation *install_path*.

*dirx_install_path*
The exact name of the root of the directory where DirX programs and files are installed. The default installation directory is *userID_home_directory*/**DirX** on UNIX systems and **C:\Program Files\DirX** on Windows systems. During installation the installation directory can be specified. In this manual, the installation-specific portion of pathname is represented by the notation *dirx_install_path*.

*dxi_java_home*
The exact name of the root directory of the Java environment for DirX Identity. This location is specified while installing the product. For details see the sections "Installation" and "The Java for DirX Identity".

*tmp_path*

The exact name of the tmp directory. The default tmp directory is /tmp on UNIX systems. In this manual, the tmp pathname is represented by the notation *tmp_path*.

*tomcat_install_path*
The exact name of the root of the directory where Apache Tomcat programs and files are installed. This location is defined during product installation.

*mount_point*
The mount point for DVD device (for example, **/cdrom/cdrom0**).

# 1. Managing DirX Identity Connectivity

Managing DirX Identity Connectivity consists of the following tasks:

- Managing connected directories
- Managing Provisioning workflows
- Managing passwords
- Managing DirX Identity servers
- Managing the Connectivity system

The next sections give a brief overview of each of these tasks and provide introductory information about Java-based and Tcl-based DirX Identity component and Provisioning workflow configuration objects.

## 1.1. Managing Connected Directories

A connected directory represents all necessary information to access source or target systems correctly.We use the term **connected directory** for DirX Identity Connectivity and the term **target system** for DirX Identity Provisioning (see the *DirX Identity Provisioning Administration Guide* for more information).

Connected directory management consists of these tasks:

- Setting up and maintaining the connected directory structure
- Setting up and maintaining the authentication information (bind profiles)
- Setting up and maintaining the schema and attribute configuration

Administrators carry out these tasks with the DirX Identity Manager.

### 1.1.1. Managing Provisioning Workflows

Provisioning workflows load data from source systems or update identities in source systems.They can also perform initial load, synchronization and validation on target systems.

Provisioning workflow management consists of these tasks:

- Setting up and maintaining the scenario structure
- Copying, changing and deleting workflows (including jobs for Tcl-based workflows)
- Creating, changing and deleting schedules

Administrators carry out these tasks with DirX Identity Manager.

Note that you can copy workflows as a sub-task of the target system wizard when creating a new target system.See the *DirX Identity Provisioning Administration Guide* for more information.

## 1.2. Managing Passwords

The relevant topic for password management in the Connectivity views group is password synchronization.

Password synchronization consists of these tasks:

- Setting up and maintaining the password scenario structure
- Setting up and maintaining the password workflows

Administrators carry out these tasks with the DirX Identity Manager.

Note that you can copy workflows as a sub-task of the target system wizard when creating a new target system.See the *DirX Identity Provisioning Administration Guide* for more information.

## 1.3. Managing DirX Identity Servers

DirX Identity servers handle all automated tasks within the Identity environment.Server management tasks include:

- Installing servers
- Configuring and maintaining servers and services, including auditing and logging
- Setting up maintenance scripts

Administrators carry out these tasks with DirX Identity Manager or with operating system-specific tools.

You can also set up distributed environments for DirX Identity servers to distribute the DirX Identity processing workload.Managing a distributed environment includes the following tasks:

- Distributed installation of components; for example, servers
- Maintaining the distributed Identity system

Administrators carry out these tasks with DirX Identity Manager or with operating system-specific tools.

## 1.4. Managing the Connectivity System

You may need to perform some additional tasks to run the Connectivity system with the Provisioning system.These Connectivity system management tasks include:

- Managing administrative accounts
- Setting up and maintaining the features that guarantee data security, including key management for data encryption and signature as well as the creation of secure connections

Administrators carry out these tasks with the DirX Identity Manager or with operating system-specific tools.

## 1.5. About Java-based Configuration Objects

Java-based configuration objects apply to the operation of all DirX Identity components that are relevant for Java-based Provisioning workflow operations.The following figure shows the main configuration objects for Java-based Provisioning workflows and their most important relationships.

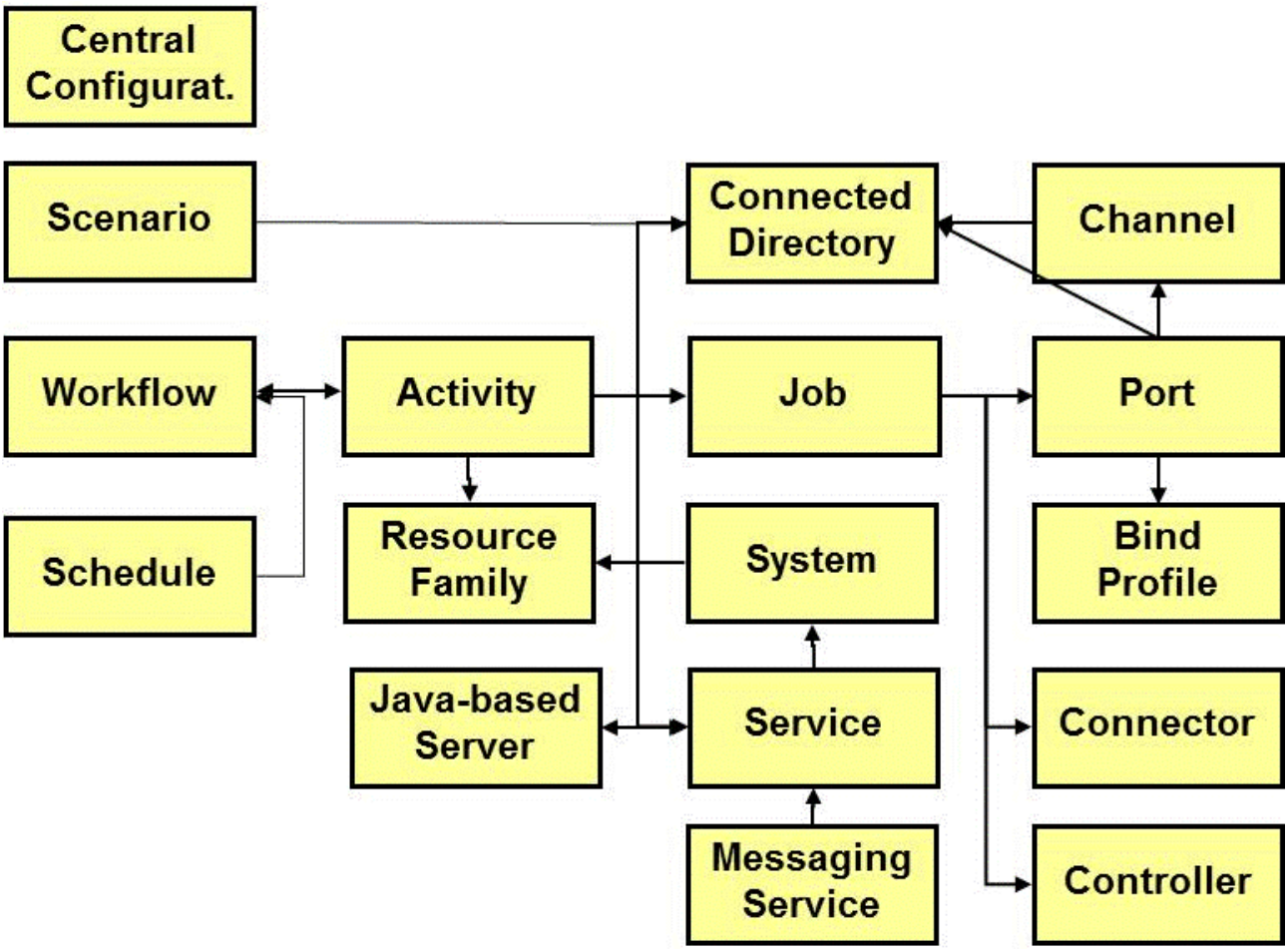



*Figure 1. Java-based Configuration Objects and their Relationships*

The description of these objects given in the context-sensitive help for DirX Identity provides information about the standard properties assigned to the object that are supplied with DirX Identity.Some of the configuration objects - for example, connected directories - can be customized to support additional customer-specific properties.

Java-based configuration objects contain information about:

- Scenario configuration objects: combine sets of connected directories and synchronization workflows and represent them graphically for a better overview and easier configuration.
- Connected directory configuration objects that represent either source or target system instances including the Identity Store. Bind profiles hold the user and password

information for a specific target system access. If you assign the connected system to a Java-based server, then this server runs the Java-based provisioning and import workflows.

- Java-based workflow configuration objects that consist of activities: procedures that transfer data from source to target connected directories.
- Java-based activity configuration objects: process steps that are combined to make a workflow. A Java-based workflow is built from a join and an error activity. The join activity is responsible for synchronizing the data between two connected systems. The error activity in event-triggered workflows receives just the entries that could not be provisioned and typically sends an email with the error information. The type of provisioning workflow and how the data are synchronized is determined by the controller configured in the join activity.
- Java-based job configuration objects are not visible at the user interface level but they exist as part of the XML configuration files. They implement the activity; for example, a join operation or error handling.
- Port configuration objects represent the access to a connected system for a workflow. Such a connected system might not only be a target system, but also the message broker and an email server. The port especially configures the connector to access the connected system. A provisioning port combines several channel configuration objects for all the object types that must be provisioned in or from the connected system.
- Java-based channel configuration objects. A channel represents an object type or an object-to-object relation for a provisioning workflow to a connected system. Typical channels are for accounts and groups and for account-group memberships. A channel describes how to find objects of this type in the connected system, which attributes to read and write, and how to map them from the other connected system.
- Schedule configuration objects that define when to run workflows.

These configuration objects refer to other configuration objects in the central configuration folder:

- Connector type objects and the XML files that describe their presentation at the user interface level.
- The Java-based Identity Servers (IdS-J servers) that run the workflows.
- The server objects that refer to service and system objects to define important information for access.
- The messaging service that allows for transferring Java Messaging Service (JMS) messages between DirX Identity components.
- Resource family objects that allow for distributing provisioning activities over several Java-based Identity servers.
- Topic configuration objects that define JMS message types.

Note: Some of these objects are marked with a red border and the text "This object might be shared because it belongs to the Configuration folder".Be careful when editing these objects because your changes can affect other objects, too.For details about the Configuration folder, see the topic "(Central) Configuration" in the context-sensitive help.

# 1.6. About Tcl-based Configuration Objects

Tcl-based configuration objects apply to the operation of all DirX Identity components that are relevant for Tcl-based Provisioning workflow operation.The following figure shows the main configuration objects for Tcl-based Provisioning workflows and their most important interrelationships.

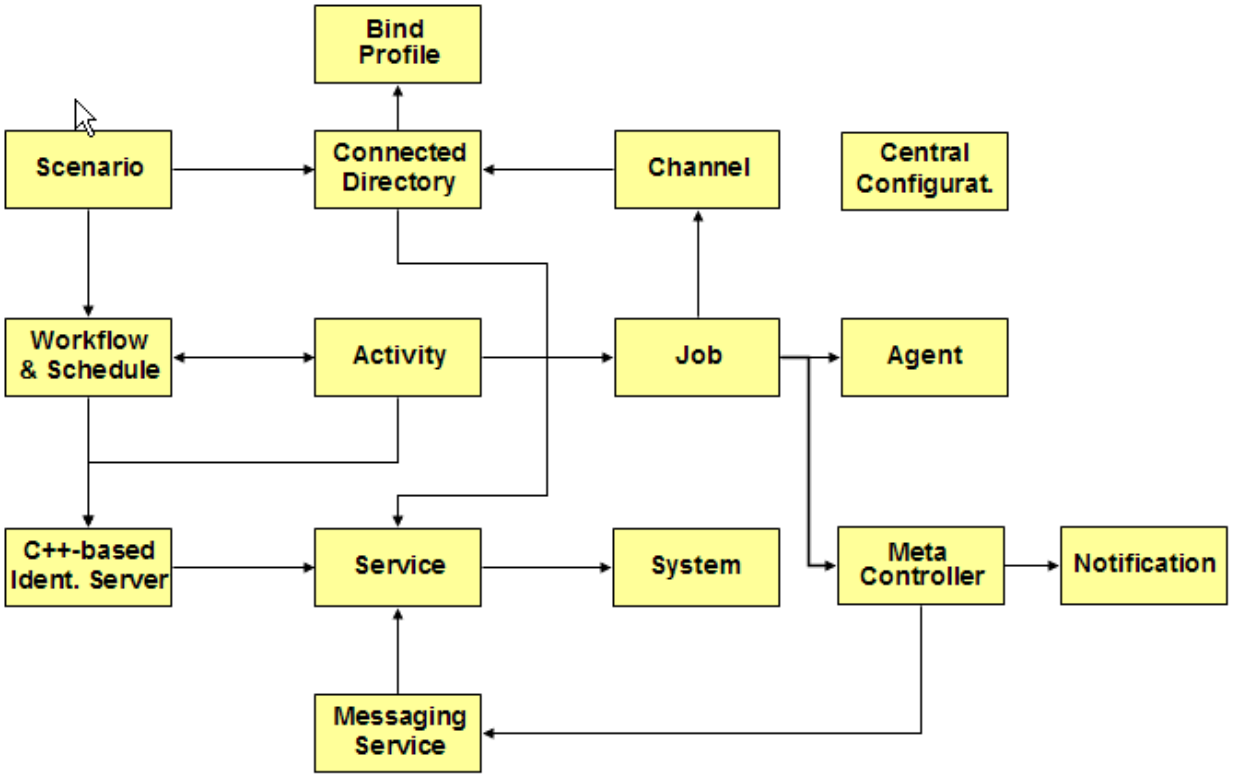



*Figure 2. Tcl-based Configuration Objects and their Relationships*

The description of each configuration object given in the context-sensitive help for DirX Identity provides information about the standard properties assigned the object that are supplied with DirX Identity.Some of the configuration objects - for example, connected directories and jobs - can be customized to support additional customer-specific properties.

Tcl-based configuration objects contain information about:

- Scenario configuration objects: specific sets of connected directories and synchronization workflows used for Tcl-based workflows.
- Connected directory configuration objects that represent either source or target system instances including the Identity Store. Bind profiles hold the user and password information for a specific target system access.
- Tcl-based workflow configuration objects that consist of activities: procedures that transfer data from source to target connected directories.
- Tcl-based activity configuration objects: sequential process steps that are combined to make a workflow. Tcl-based activities can represent jobs or complete workflows.

- Tcl-based job configuration objects: fully configured agents that can be used by Tcl-based activities in workflows.
- Agent configuration objects defined by an agent type: un-configured programs or procedures that are called by a job. The meta controller acts as special agent and represents the join engine.
- Tcl-based channel configuration objects.
- Schedule configuration objects that define when Tcl-based workflows are to run.

All these configuration objects access specific configuration in the central configuration folder:

- Agent type objects and the XML files that describe their presentation at the user interface level.
- All central notification definitions used by meta controller jobs.
- Definitions for standard files that are recognized by the agent controller.
- Central Tcl files used by meta controller jobs.
- C-based Identity Servers (IdS-C servers) - the DirX Identity component that must exist on each host server on which an agent or a C connector is to run.
- The server objects that refer to service and system objects to define important information for access.
- The messaging service that allows for the transfer of JMS messages between DirX Identity components.



Some of these objects are marked with a red border and the text "This object might be shared because it belongs to the Configuration folder". Be careful when editing these objects because your changes can affect other objects, too. For details about the Configuration folder, see the topic "(Central) Configuration" in the context-sensitive help.

# 2. Planning for DirX Identity Connectivity

This chapter provides a planning checklist of the issues that you must take into account when deciding how to deploy DirX Identity Connectivity in your environment.

Planning tasks include:

- Identify the source systems
- Decide on source system workflows
- Identify the target systems
- Decide on target system workflows
- Decide on password synchronization
- Define the system architecture
- Define system maintenance

## 2.1. Identify the Source Systems

- What are the source systems that create and maintain identity information?
- What are the characteristics of the source systems in terms of authentication and authorization?
- What is the best connected directory template that can be used for source system integration?

## 2.2. Decide on Source System Workflows

- What is the best workflow template that can be used for source system integration?
- Is it necessary to set up secure connections?

## 2.3. Identify the Target Systems

- What are the target systems that must be integrated?
- What are the characteristics of the target systems in terms of authentication and authorization?
- How should these target systems be migrated into the DirX Identity Connectivity environment?Can there be an initial load of accounts and/or groups into DirX Identity?
- Is there a need for virtual target systems?
- What is the best connected directory template that can be used for target system integration?

## 2.4. Decide on Target System Workflows

- Which technology shall be used: Java-based real-time workflows or Tcl-based

technology or both?

- What is the best workflow template that can be used for target system integration?
- Is it necessary to set up secure connections?

## 2.5. Decide on Password Synchronization

- Which target systems shall be synchronized with the central password solution?
- What are the possible sources for password changes: the Windows login, a web solution or both?
- Shall the messaging connections be secured via SSL?

## 2.6. Define the System Architecture

- How many Identity domains shall be handled?
- Shall all Identity domains be handled by separate Connectivity domains or do you plan to handle all Identity domains by a single Connectivity domain?
- Which connections should be secured, for example via SSL?
- Which attributes are required to be encrypted?
- Shall your administrative (bind) passwords be encrypted?

## 2.7. Define System Maintenance

- How often shall your administrative passwords be changed?
- Define a procedure to change your administrative passwords consistently.
- What is the interval at which to change your certificates for data and administrative password encryption?
- Define a procedure to change your certificates from time to time.

# 3. Managing Connected Directories

Connected directory management tasks include:

- Setting up and maintaining the connected directory structure
- Setting up and maintaining the authentication information (bind profiles)
- Setting up and maintaining the schema and attribute configuration

The next sections discuss these tasks in more detail.

## 3.1. Setting Up the Connected Directory Structure

A connected directory represents a single data store in an Identity environment.The connected directory configuration object holds all the data required to describe the properties of the respective directory or database that is common for access by all workflows or services.

Connected directories can reside under the connected directories folder (or one of its subfolders) or under a job object.In the latter case, the connected directory is an intermediate connected directory that is used to store information between two related activity steps of a workflow.

The folder structure under the connected directory folder depends on the scenario structure you intend to create.If you use the Provisioning view's target system wizard, the wizard creates the structure automatically.If the wizard-generated structure is not correct, you can create your own folders and move objects to them accordingly.Be aware that the target system wizard creates additional objects in the default folder structure.You must move these objects after using the wizard.

For source system provisioning, you must create and maintain your own scenario and folder structure.You can include the same connected directory instance into several scenarios (use the **Assign** menu item) to separate specific workflow groups from each other.

## 3.2. Setting Up Connected Directory Authentication Information

Connected directories allow you to define the authentication information via bind profiles.A connected directory can have one or more bind profiles that can be used by several workflows.

Bind profiles let you define the user and password information, where the password can either be stored in a simple scrambled format or in encrypted format.Be aware that the scrambled format is simply not readable but nevertheless not secure at all.It is easy to crack.To protect the information in your connected directories, we recommend using encrypted storage of bind profile passwords.

Bind profiles also define the security level for all access information (for example SSL/TLS).

## 3.3. Schema and Attribute Configuration Handling

Connected directory configuration objects can contain schema information, but only the information that is necessary to configure the synchronizations properly.This type of schema information consists mainly of the directory objects and their related attributes, which are necessary for attribute selection and mapping.

DirX Identity requires the schema information to be in a specific "attribute configuration" format that is mainly used by the meta controller.This format allows for the description of the schema information required for any directory type.Supported directory types are:

- LDAP directories with a flexible and extensible schema (for example, DirX and Active Directory).
- Other databases (for example, ODBC) with a flexible and extensible schema
- Databases and directories with a fixed schema (for example, the Windows NT directory)
- File directories, which keep a collection of files in the same format.

### 3.3.1. Schema Handling for LDAP Directories

For LDAP directories, the DirX Identity Manager (at the administrator's request) can read the relevant part of the schema directly from the directory.

Administrators must explicitly update this schema information with DirX Identity Manager after making schema changes or extensions. Note that a schema read from Active Directory requires the presence of a bind profile in the connected directory configuration object.

After a schema update, DirX Identity generates the attribute configuration information for the LDAP directory automatically from the schema information after requesting confirmation by the user.

DirX Identity provides a comprehensive mechanism to customize the LDAP schema update. See "Using the Schema Displayer" in the chapter "Using DirX Identity Manager" in the *DirX Identity User Interfaces Guide*.

### 3.3.2. Schema Handling for Other Databases

For ODBC databases or other similar types, the administrator uses the DirX Identity Manager to enter the schema information into the configuration database. The administrator uses the Attribute Configuration Editor to enter the information by hand or uses the editor's import selection to import the information from an existing attribute configuration file.

A schema configuration object is not required for these kinds of directories. The attribute configuration information is sufficient.

### 3.3.3. Schema Handling for Fixed Directories and Databases

Some databases-for example, Windows NT - have a fixed schema. DirX Identity stores these schemas in the central configuration object underneath the connected directory type configuration object. A reference points from the connected directory instance to these entries.

### 3.3.4. Schema Handling for Files

Connected directories can also be a collection of files with the same format; that is, the same schema description. If you want to model files with a different schema description, you need a separate connected directory definition for each file.

This common description is held in the attribute configuration object. No schema object is necessary. In contrast to the information for LDAP directories, additional information for field, record and multi-value separators is necessary.

The administrator uses the Attribute Configuration Editor in the DirX Identity Manager to enter this information by hand or by an import from an existing attribute configuration file. For details about the Attribute Configuration Editor and DirX Identity Manager, see the *DirX Identity User Interfaces Guide*.

# 4. Managing Provisioning Workflows

This chapter discusses how to manage:

- Java-based Provisioning workflows
- Tcl-based Provisioning workflows

## 4.1. Managing Java-based Provisioning Workflows

This section provides information about managing Java-based Provisioning workflows for real-time synchronization, including:

- Runtime operation
- Target system Provisioning workflow operation
- Error handling and retry
- Cluster workflows
- Target system attributes for cluster workflows
- How to copy Java-based Provisioning workflows
- How to start Java-based workflows

For trouble shooting hints in a real-time workflow scenario see the section "Real-time Synchronization" in the *DirX Identity Troubleshooting Guide*.

### 4.1.1. Java-based Workflow Runtime Operation

This section describes Java-based workflow runtime operation, including:

- The general concept of runtime operation for real-time (event-based) and scheduled workflow execution
- Java-based Server support for real-time event handling and message flow

#### 4.1.1.1. General Runtime Operation

The following figure illustrates the general concept of runtime operation for a Java-based workflow for real-time synchronization:

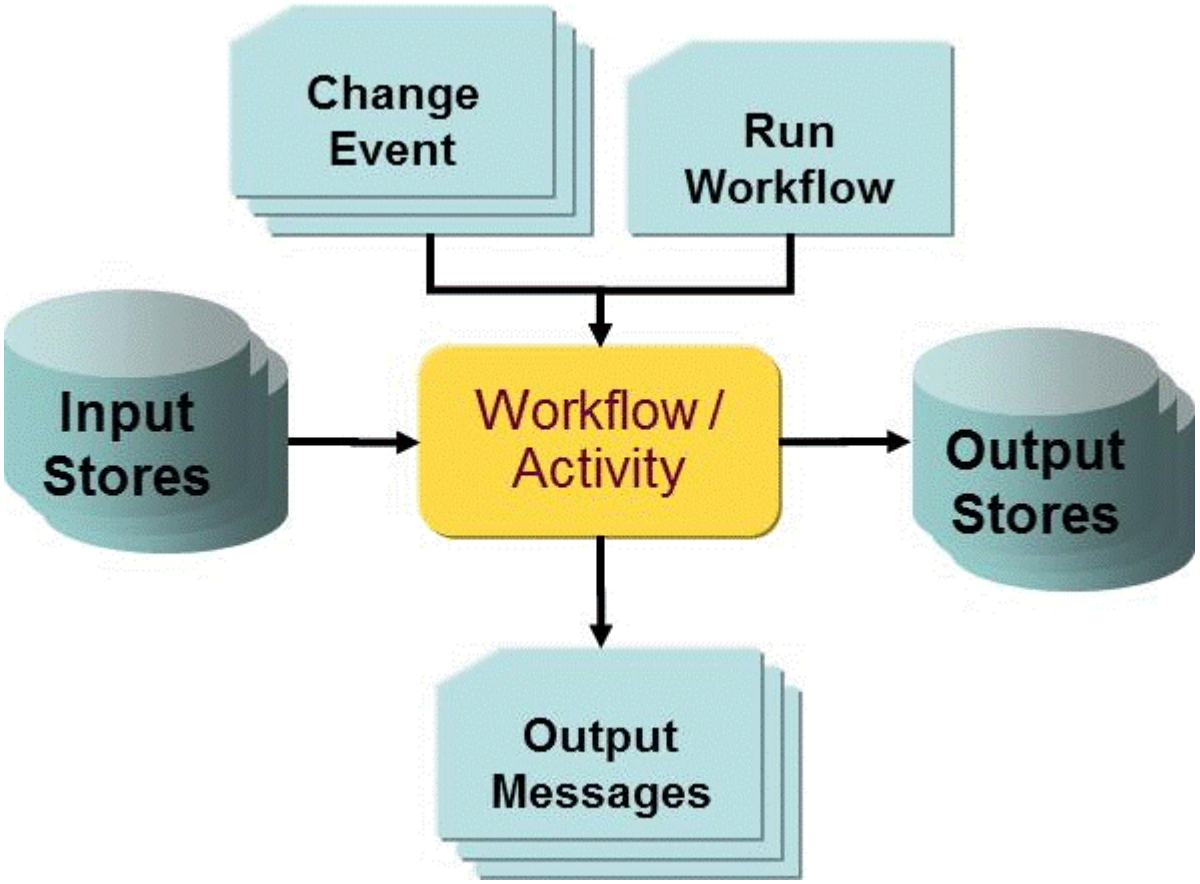



*Figure 3. Java-Based Workflow Runtime Operation*

Java-based workflows are started by messages. These can convey an entry change, a request to change a password or a request to start a workflow.

An important example is the change of an account or a group. In this case, the event contains the DN of the changed entry within the input store, typically the DirX Identity domain. The workflow reads this object from the input store, transforms it into the format of the output store and performs the necessary operation at the output store. If necessary, it can create additional messages to trigger other real-time workflows.

You can also start Java-based workflows at a specific time via schedules or immediately using DirX Identity Manager. In this case, the workflow works on all objects in the input store that are identified by the search filters in the corresponding channels.

The general runtime concept can be used in a variety of applications.

#### 4.1.1.2. Server Support for Runtime Operation

The following figure illustrates the logical structure of the Java-based DirX Identity server components.

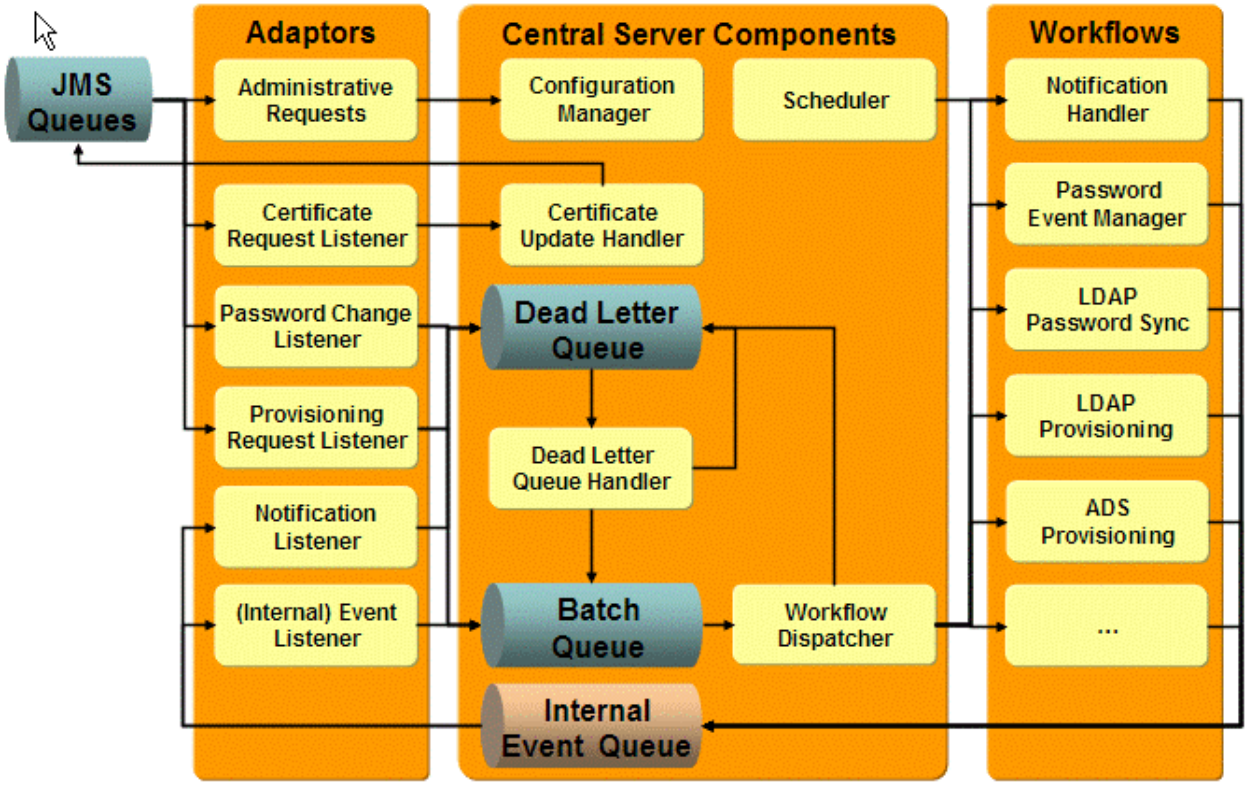



*Figure 4. Logical Server Structure*

The Java-based Server contains two workflow engines: one for real-time provisioning workflows and one for request workflows (see the *DirX Identity Provisioning Administration Guide* for more information about request workflows).

Provisioning workflows are started by events. The Java-based Server uses functions called adaptors to read events from Java Messaging Service (JMS) queues.

The Java-based Server can handle the following types of events (not all of them are shown in the figure):

- **Administrative Requests** (**Admin Request Handler**) - currently only the **Load IdS-J Configuration** request from the DirX Identity Manager is handled.
- **Configuration Requests** (**Configuration Handler**) - the Windows Password Listener (or any other component that needs a current certificate) requests the initial certificate and a list of available messaging services. The **Configuration Handler** sends the requested information via a JMS message to the requester (broadcast). If update requests cannot be handled, the event is put into the **Dead Letter Queue**.
- **Notification Requests** (**Mail Listener**) - all components can send notification requests. These requests are centrally handled by a Notification Handler, which sends the messages to the mail server. If messages cannot be sent, the event is put into the **Dead Letter Queue**.
- **Real-time Events** - various listeners read events for specific topics and put them into the **Batch Queue** for further processing. After delivery to this queue, the JMS event is confirmed and deleted from the JMS queue. If an event does not have the correct format, it is put into the **Dead Letter Queue**.

Real-time event listeners are:

- **Account Password Change Listener** - reads all account password change events.
- **Entry Change Listener** - reads all events resulting from entry changes.
- **Password Change Listener** - reads all user password change events.
- **Provisioning Request Listener** - reads all provisioning request for connected systems.
- **SetAccountPasswordListener** - reads password events from the User Password Event Manager to update accounts.

- **Start Workflow Requests** - reads all requests for starting event-based processing and provisioning workflows listeners. Start workflow requests listeners and adapters are:
  - **Entry Change Start Workflow Listener** - reads all requests for starting event-based processing workflows, automatically collocated with the Entry Change Listener.
  - **Provisioning Request Start Workflow Listener** - reads all requests for starting provisioning workflows, automatically collocated with the Provisioning Request Listener.
  - **Resolve User requests** – resolves the assignments of the selected user to the access rights in the target systems. This function is performed by the Resolution Adapter.

For efficient processing, events of the same type are bundled as batch requests if requests are arriving at a high rate. The **Workflow Dispatcher** reads these batch requests and assigns them to already running but waiting **Workflow Threads**. If it is not possible to find a suitable workflow thread, the Workflow Dispatcher places the request in the **Dead Letter Queue**.

The **Workflow Threads** try to resolve the requests. If successful, they send a response message to the **Workflow Dispatcher** indicating that the task was successful and the request is deleted from the queue. If the request cannot be resolved after a configurable number of retries, the Workflow Threads place it in the **Dead Letter Queue**.

Workflow Threads can create additional events. For example, the **Password Event Manager** creates a set of **Password Synchronization Requests** for the related target systems and stores them in the **Internal Event Queue**. The (Internal) Event Listener reads these requests, bundles them as batch requests and places them in the **Batch Queue**.

Requests stored in the **Dead Letter Queue** are not handled automatically again because normally administrative action is necessary before further processing is worthwhile. An administrator can

- Solve the related problem (for example, there was no workflow set up for specific request types or the target system was not available). He can then process the stored events again. The **Dead Letter Queue Handler** bundles them as batch requests and puts them into the **Batch Queue**.
- Decide to delete stored requests if further processing no longer makes sense (for example, because the parameters of the requests are wrong).

### 4.1.2. Target System Provisioning Workflow Operation

This section explains the general concept of operation for the real-time Provisioning workflows between the Identity Store and target systems. The following figure illustrates how request and real-time (Java-based) workflows cooperate to process a user creation request.

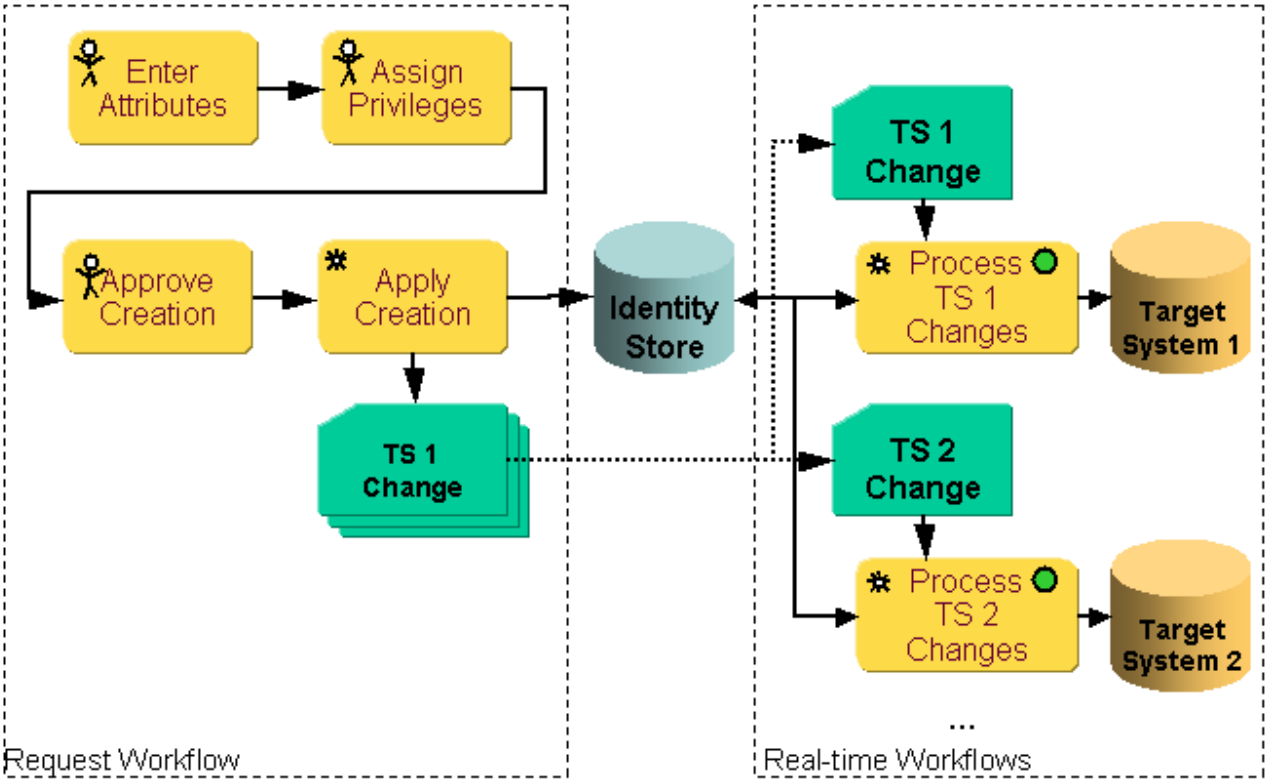



*Figure 5. Request and Real-Time Workflow Interaction*

The example shown in the figure assumes that a request workflow for user creation with several interactive activities results at the end in the automatic activity Apply Creation:

- The **Apply Creation** activity creates the user entry and calculates based on the assigned privileges during the privilege resolution the necessary set of accounts and group memberships in various target systems. This information is written into the Identity Store and in parallel sent as events to the JMS messaging service (TS *n* Change).
- The corresponding **adaptor** (Provisioning Request Listener) reads the events from an external queue. It puts the events into the Internal Event Queue (permanent queue) and deletes them from the external queue.
- A **workflow dispatcher** component (not shown in the figure) analyzes the event in the Internal Event Queue and starts the workflows that can handle these types of events. In this case, the relevant target system Provisioning workflows are started.
- After start, the **target system workflow** analyzes the event's content (the event contains typically only the DN of the relevant object):
- The target system workflow reads the object from the Identity Store, performs mapping and join operations and writes it to the target system. If this operation is successful, the status of the relevant object in the Identity Store is updated accordingly.

- The workflow engine retries unsuccessful requests that are the result of temporary errors (for example, the network was temporarily unavailable) until the retry limit is reached. It then passes the requests and their responses to the error activity, which issues e-mail notifications. The failed request is stored in the Java-based Server's dead letter queue, where administrators can evaluate it and perform another retry operation once the cause of the error has been eliminated.
- The target system workflows write logging, auditing and statistics information to be processed by the corresponding handlers.
- At the end of each workflow, successful or not, the requests are removed from the Internal Event Queue.

### 4.1.3. Error Handling and Retry Operation

Each activity within a Java-based workflow has two channels that the Java-based Server's real-time workflow engine uses to combine unsuccessful requests with their responses: an error channel and a retry channel. The choice depends on the error code that the connector returns in its SPML response.

- An error code FAILURE indicates a permanent error. This type of error typically results from incorrect bind credentials or a non-existent base node for searches. The Java-based Server's real-time workflow engine places permanent error SPML responses into the error channel.
- An error code TEMP_ERR indicates a temporary error. This type of error typically results from a lost connection. The Java-based Server's real-time workflow engine places temporary error SPML responses into the retry channel.

After a configurable waiting period, the workflow engine restarts the activity with the requests in the retry channel. If the configured retry limit is reached, it puts the requests and error responses into the error channel.

If an error activity is configured, it receives the requests and responses from the error channel and notifies the affected user with an e-mail.

If this fails or if no error activity is configured, the real-time workflow engine places the request in the Java-based Server's dead letter queue. Administrators can use the Java-based Server's Web Admin interface to view the failed requests and responses in this queue and to delete or re-process them. For details about dead letter queue handling, see the "Server Support for Runtime Operation" section. For details about Web Admin, see the chapter "Using Web Admin" in the *DirX Identity User Interfaces Guide*.

An activity usually processes a batch of requests. The entire activity is classified as failed only when one of its components is unable to process any request. This means that the workflow is classified as successful when some requests have temporarily failed or when the error activity has successfully handled permanent errors.

### 4.1.4. Using Cluster Workflows

Typically, you set up one workflow type for provisioning from a target system to a connected system. Setting up many similar target systems of the same type requires you to

set up the corresponding number of workflows. Changing the behavior of these workflows can be cumbersome because you need to adapt all these definitions by hand. If the properties of the connected systems are similar (except for the address and bind information and a few specific properties), you can use **cluster workflows**, which allow you to set up one workflow that is controlled by properties of the target system instances. Using cluster workflows allows you to set up and maintain a single workflow (or one set of workflows for synchronization, validation and password synchronization) for a cluster of target systems. The following figure illustrates the cluster workflow concept.

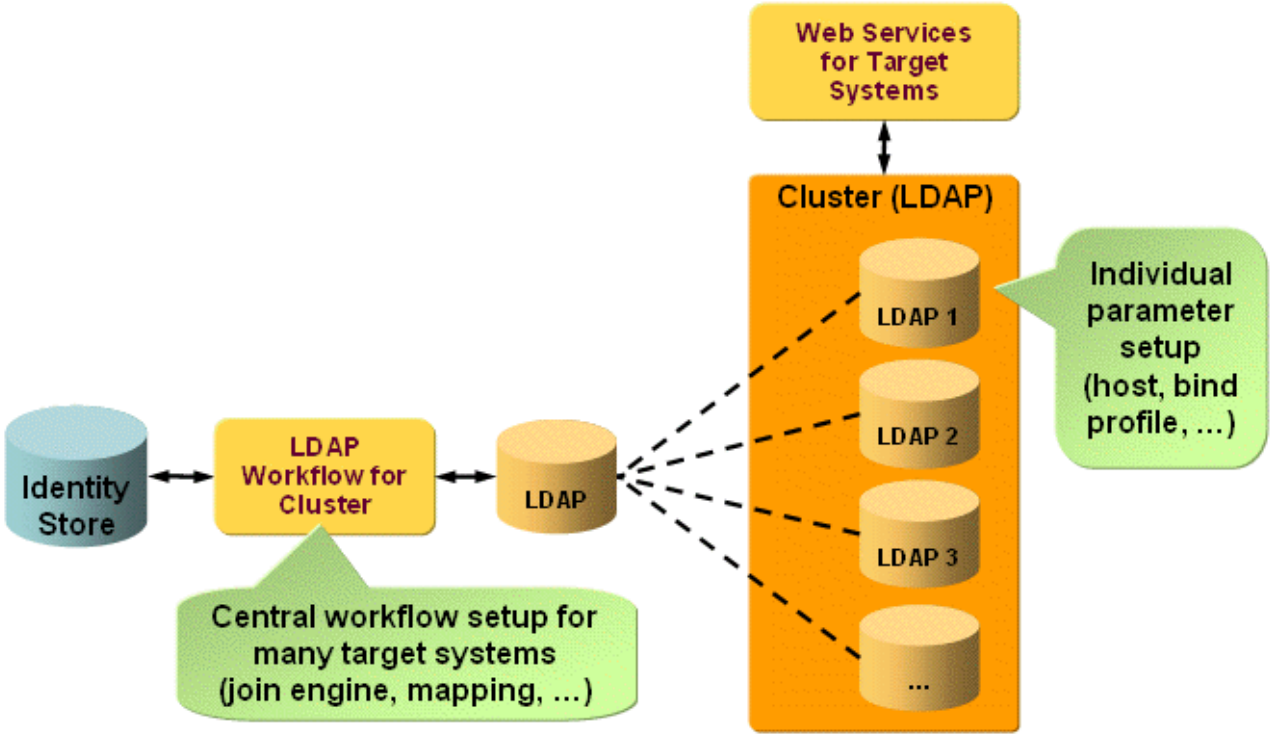



*Figure 6. Cluster Workflow Operation*

As shown in the figure, you can set up one workflow that handles the entire set of clustered target systems. To use this method, the target systems must have identical configurations regarding most of the parameters (for example, attribute mapping).

The next sections describe the cluster workflow concept in more detail, including:

- A comparison of non-clustered and clustered solution configurations
- The individual parameters to access the target system of the cluster are defined at the target systems in the Provisioning view group. Typical parameters are host, port and bind profile.
- The workflows can run in event-based or scheduled mode. For schedule setup information, see the section "Scheduling Cluster Workflows".

#### 4.1.4.1. Setting Up Non-clustered and Clustered Workflows

This section explains the differences between the non-clustered and clustered setup of Java-based workflows. The following figure illustrates the non-clustered workflow and target system configuration:

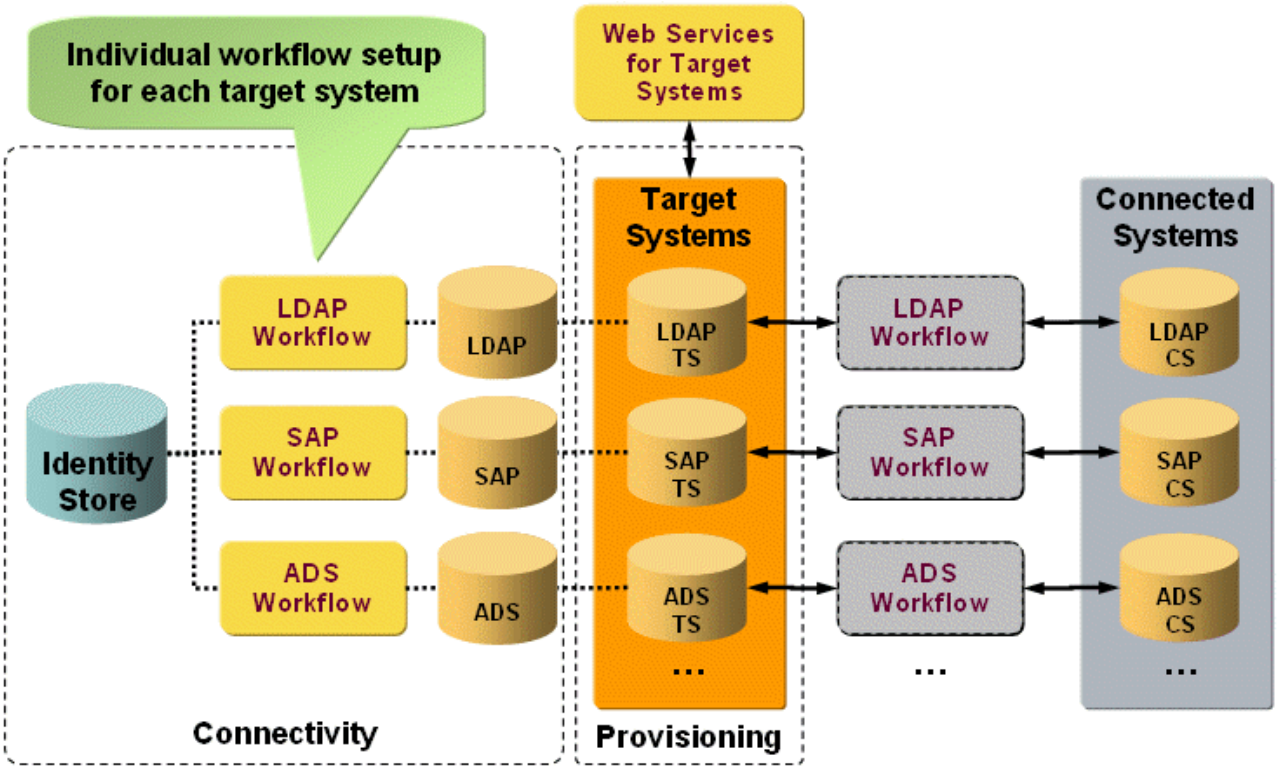



*Figure 7. Non-clustered Solution*

To set up a non-clustered solution, set up one workflow definition between the connected directory **Identity Store** and each connected directory that represents your connected system (here: LDAP, SAP, ADS). Set up each workflow independently. When the workflow subsequently runs between the target system and the connected system, it takes all configuration information from the workflow definition and the connected directory and all data from the target system and/or from the connected system.

The following figure illustrates the clustered workflow and target system configuration:

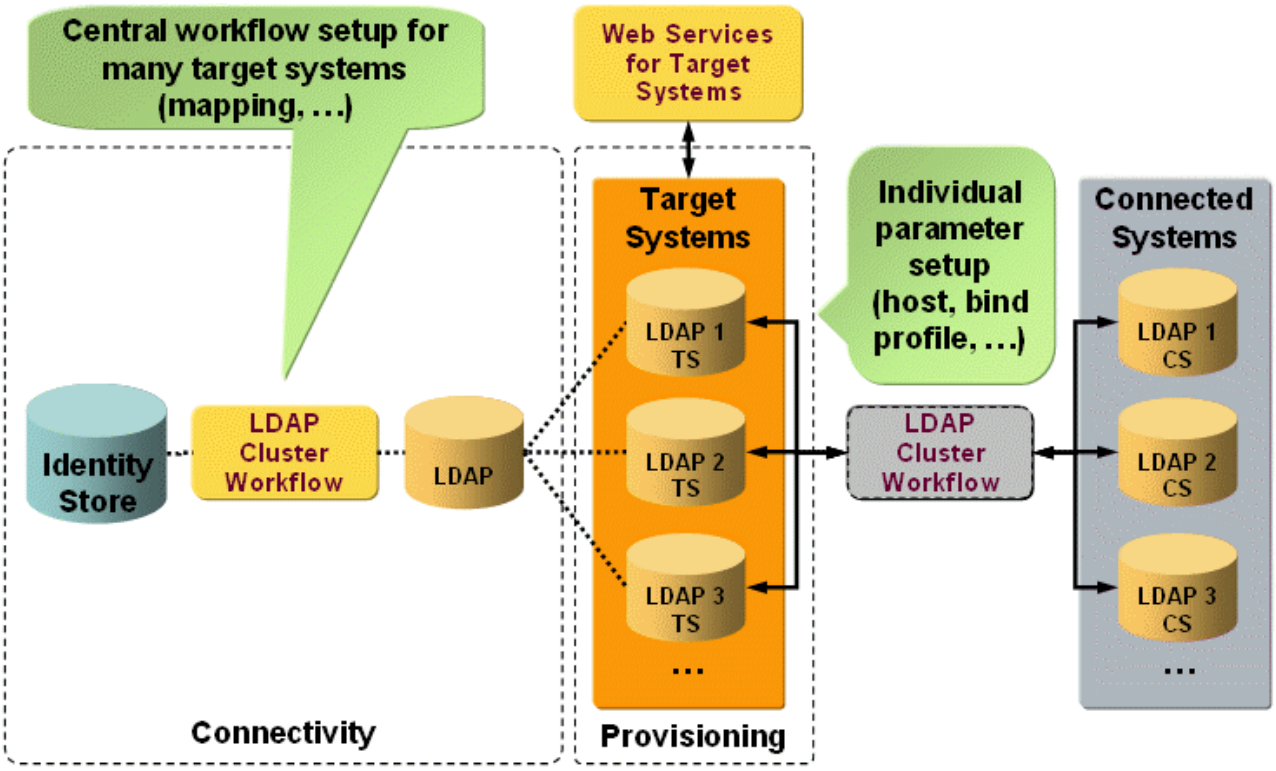



*Figure 8. Clustered Solution*

To set up a clustered solution, set up one workflow definition between the Identity Store connected directory and one connected directory that represents all instances of your connected system (in this example, LDAP). The workflow setup is valid for all systems in the cluster. When the workflow subsequently runs between the target systems and the connected systems, it takes the static part of the configuration information from the workflow definition and the dynamic part from the target system definition and from the **Workflow Configuration**, **Connector Configuration** and **Environment Properties** tabs.

Depending on the data to be transferred from right to left or left to right, the correct individual parameters are evaluated by the workflow instance and used for accessing the connected system.

#### 4.1.4.2. Configuring Cluster Workflows

Cluster workflows read their target system-specific options only from the target system in the DirX Identity domain. While other workflows read these options from the connected directory, this is not possible for a workflow that supports a number of target systems.

Therefore, the target system allows for setting a number of new attributes for the workflow configuration. The target system provides a set of standard attributes and a "specific attribute" feature to configure additional attributes without the need for extending the LDAP schema. These configuration options can be separated into workflow configuration attributes, connector configuration attributes and environment configuration attributes. The next sections provide more information about these attributes.

##### 4.1.4.2.1. About Workflow Configuration Attributes

Workflow configuration attributes are a set of fixed properties that contain the address and

bind information to access the connected system. Since most connectors share a set of common configuration parameters, DirX Identity has defined a standard set of options: address, data port and secure data port, SSL, authentication mode, path to key store and trust store files and key store alias name.

For binding of the target system connector, you must reference an account in the domain. The attributes "dxrName" and "dxmPassword" of this account are used as user name and (encrypted or scrambled) password.

#### 4.1.4.2.2. About Connector Configuration Attributes

Connector configuration attributes define additional target system-specific configuration options for the target system connector. These attributes must be defined as environment properties in the Connector Configuration tab and are stored as tag/value pairs in the LDAP attribute "dxmSpecificAttributes".

For example, if you want to specify some connection-specific properties of a connector, like the following connection section properties for the SapR3UMConnector:

General:

```
<property name="client" value="800"/>
<property name="language" value="en"/>
```

For connecting without load-balancing:

```
<property name="systemID" value="08"/>
```

For connecting via gateway :

```
<property name="systemIDgateway" value="08"/>
<property name="gwhost" value="host"/>
<property name="gwserv" value="server"/>
```

For connecting via load-balancing (messaging service):

```
<property name="r3SystemName" value="sysname"/>
<property name="group" value="group"/>
```

Other parameters, for example, access to CUA:

```
<property name="accessToCUA" value="FALSE"/>
```

```
<property name="combinedRoleProfileSubsystem" value="FALSE"/>
```

For more options, see the content of object “ts” (section connection) in a configured real-time SAP workflow.

Specify them as tag/value pairs (for example, "client/800") in the Connector Configuration tab of the provisioning target system object.

#### 4.1.4.2.3. About Environment Configuration Attributes

Environment configuration attributes define options for the workflow mapping. Since these options vary significantly from workflow to workflow, there are no standard attributes defined. Instead, they are all stored in the LDAP attribute "dxrEnvironmentProperties" as tag/value pairs. Parameters that are used in many workflows include account and group base both in DirX Identity and in the target system, domain name, "no_members" for the default group member if creating a group without account members. As a default rule, take all specific attributes of the connected directory and store them in the target system as environment properties.

Note that some logically equivalent attributes were entered both in the target system and the connected directory: the base nodes for accounts and groups in DirX Identity. The attributes of the connected directory are used in workflow configuration, the attributes in the target system are referenced in the object descriptions when creating accounts and groups. The latter ones are stored as tag/value pairs in the LDAP attribute "dxrOptions". The parameters for workflow configuration are now stored in "dxrEnvironmentProperties". You can avoid this duplicate storage by either changing the object descriptions of accounts and groups or by changing the workflow configuration, typically the export section and the identifier mapping.

### 4.1.4.3. Scheduling Cluster Workflows

If you use the scheduler, you can set up additional parameters in the tab **Target System Cluster**. Set the **Search base** and **Filter** parameters to define the set of target systems the schedule shall handle.

## 4.1.5. Copying Java-based Provisioning Workflows

You can copy Java-based workflows either in DirX Identity Manager’s Global View or in its Expert View. The next sections describe how to perform these tasks. For details about DirX Identity Manager, see the chapter "Using DirX Identity Manager" in the *DirX Identity User Interfaces Guide*.

### 4.1.5.1. Copying Java-based Workflows in the Global View

To make copy operations easier and more intuitive, DirX Identity provides a complex copy method that is activated in DirX Identity Manager using the **Configure** method for connected directories and the **New** method for workflows. Both methods work closely together.

First, you create new connected directory icons that represent your connected directories.

With the **Configure** method, you can configure these objects based on other existing connected directories that act as templates in this case. This template is copied to represent your new connected directory. Note that the related service object is also copied, which avoids interference with the original object.



To avoid copying files and channels that will never be used, the copy operation does not copy files and channels of connected directories of type **File**. Instead, the data files and channels are created when a workflow is copied. As a result, it does not make sense to create data files during such a copy operation.

When two or more of these connected directories exist, you can link these objects via workflow lines. Next, you can either link existing workflows to these lines (using the **Assign** method) or create copies of existing workflows (using the **New** method) that act as templates. This template is copied to represent your new workflow.

Copying these templates is a complex procedure that handles all types of Java-based workflows in addition to the password synchronization workflows.

- DirX Identity tries to find all workflows that fit between the two connected directories. If no workflows can be found, an error message is displayed.
- The wizard tries to identify the corresponding target system in the Provisioning view group to fill the Cluster and Domain parameters. If the target system could not be found the error message "No associated target system for *targetsystem*" is displayed. You can ignore this message (click **Yes**) and set the two parameters by hand later on.
- The workflow is copied, including all activities and ports.
- All referenced channels are copied to the related new connected directories and linked to the corresponding ports. The procedure uses existing channels where possible.
- A channel folder is created beneath the connected directory that represents the target system if -n workflows refer to it but have different Identity Stores.

Some types of workflows cannot be copied with the **New** method and thus must be copied in the Expert View:

- Password synchronization workflows with JavaScript Mapping (you can identify this type of workflows by the blue flag symbol).

Both the **Configure** method for connected directories and the **New** method for workflows keep the original folder sub structure in the scenario to which the objects are copied. You can change these structures at any time with the **Move** method in the Expert View if required.

#### 4.1.5.2. Copying Java-based Workflows in the Expert View

If you copy an object in the Expert View, the complete sub-tree is copied. For example, copying a workflow copies the workflow object and all activities with all links to other objects. Linked objects - channels, in this case - are not copied. As a result, you must copy the linked objects separately and change the links accordingly.

For example, if you copy workflow W1 with activity A1 that points to a channel C1, this results in workflow W2 with activity A1 that points to channel C1. The activity name can remain the same because it resides under a different workflow object.

In this example, you must copy channel C1 separately, which results in channel C2 with the relevant Java-mapping objects. Now you must relink activity A1 from workflow W2 to channel C2. This action makes the two workflows (W1 and W2) independent down to the channel level. If other channels are referenced after this copy, you must repeat this procedure for all of these objects.

Because this is a difficult and error-prone procedure, we recommend that you use the copy procedures provided in the Global View whenever possible.

Because password synchronization workflows have a simple structure with only a few links, it is safe to copy these workflows in the Expert View.

### 4.1.6. Starting Java-based Provisioning Workflows

You can start Java-based Provisioning workflows in two different ways:

- DirX Identity clients can send real-time events that trigger the workflows.
- You can set up schedules to run the workflows regularly at specific times or time periods.This method is especially useful for validation workflows, which are typically not triggered by events.

For more information about Java-based workflow runtime operation and the Java-based Server's real-time event handling operations, see the section "Java-based Workflow Runtime Operation".

## 4.2. Managing Tcl-based Provisioning Workflows

This section describes how to manage Tcl-based Provisioning workflows, including:

- Runtime operation
- Configuration files and Tcl scripts
- Object naming
- Linking objects
- Understanding notifications
- Tcl-based workflow design rules
- Copying Tcl-based workflows
- Starting Tcl-based workflows

### 4.2.1. Runtime Operation

This section describes Tcl-based workflow runtime operation, including:

- How DirX Identity handles the workflows and how the Tcl- and C++-based components

work together

- DirX Identity agent controller runtime tasks
- The execution status values generated during Tcl-based workflow and activity runs
- The checkpoints
- File paths and areas for Tcl-based workflow runtime operation

#### 4.2.1.1. Basic Runtime Operation

The following figure shows the interactions between DirX Identity components during a run of an example Tcl-based provisioning workflow with two activities (Active Directory export and identity store import). The example assumes that the administrator starts the workflow by hand from the DirX Identity Manager.

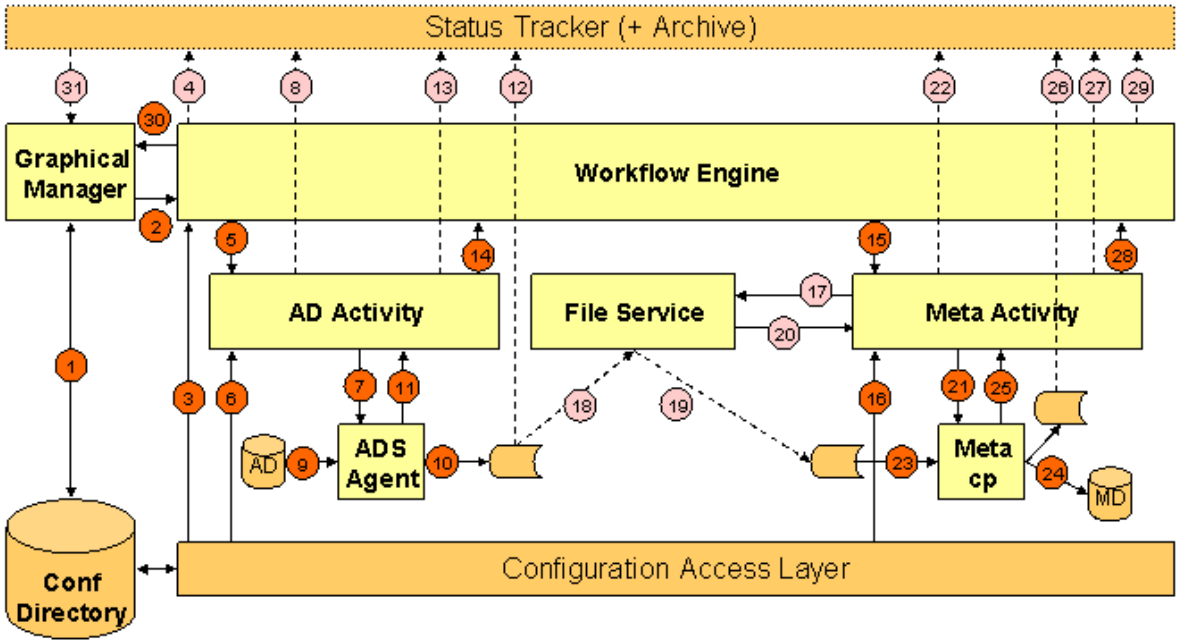



*Figure 9. Component Runtime Operation*

As shown in the figure:

1. The DirX Identity administrator configures the workflow using the DirX Identity Manager Global or Expert view.
2. The DirX Identity administrator starts the workflow from the DirX Identity Manager.
3. The workflow engine (WFE) reads the workflow configuration data (the activities to be started and the sequence in which to start them).
4. The WFE notifies the start of the workflow to the status tracker (STT). A workflow status entry is created in the configuration database.
5. The WFE starts the C++-based DirX Identity server's agent controller (AGC) to run the Active Directory (AD) activity.
6. The AGC reads the activity configuration data for the AD activity and prepares the ADS agent environment (creates the agent configuration "ini" file).

7. The AGC starts the ADS agent.
8. The AGC notifies the STT again.
9. The ADS agent reads the Active Directory according to the "ini" file configuration.
10. The ADS agent writes an intermediate data file and a trace file.
11. The ADS agent exits and reports this action to the AGC.
12. The AGC moves all required files to the status area specified by the status path.
13. The AGC notifies the STT about the completion of this activity.
14. The AGC exits and reports to the WFE. The WFE finishes the AD activity.
15. The WFE starts the AGC again to run the meta activity.
16. The AGC reads the configuration data for the meta activity.
17. The AGC starts the file service if the activities ran on different machines and if there is no shared file system configured.
18. The file service reads the file from the source location.
19. The file service copies the file to the required target location.
20. The file service exits and reports this action to the AGC.
21. The AGC prepares the meta controller environment (creates the Tcl and attribute configuration files) and starts the meta controller.
22. The AGC notifies this to the STT.
23. The meta controller reads the intermediate file.
24. The meta controller imports the information from the intermediate file into the identity store.
25. The meta controller exits and reports this action to the AGC.
26. The AGC moves all relevant files to the status area.
27. The AGC notifies the STT about this action.
28. The AGC then exits and reports to the WFE.
29. The WFE finishes the meta activity and reports this action to the STT.
30. The WFE exits and reports to the DirX Identity Manager.
31. The DirX Identity Manager can read the complete monitor information including the information in the status area. Of course you can view status information throughout the workflow run to watch the progress of the operation.

#### 4.2.1.2. Agent Controller Runtime Tasks

The C++-based Server’s agent controller component performs three main tasks during an agent run:

1. Produces the files that the agent requires. These files can be "ini" files, Tcl script files, or attribute configuration files. During this step, the agent controller resolves all references to the property settings of other objects in the configuration database that are contained in these files.

2. Runs the agent on the correct C++-based server using the command line defined in the job. The command line can also contain references (for example, to file name definitions in other objects); the agent controller resolves these references before it runs the agent.
3. Transfers the agent's exit code (through the status tracker) to the corresponding status entry after the agent terminates. Next, the agent controller copies all required files-configuration files, input files, output files, trace file, and report files-into the status area specified by the status path in the C++-based Server configuration object. (You can use the file configuration object to configure this behavior for each file; by default, all files are copied). The agent controller notifies the Status Tracker about these copy operations and the locations of the files and writes this information into the corresponding status entry. The agent controller deletes all files in the work area that the administrator has defined to be temporary in the configuration database.

The following figure illustrates these tasks.

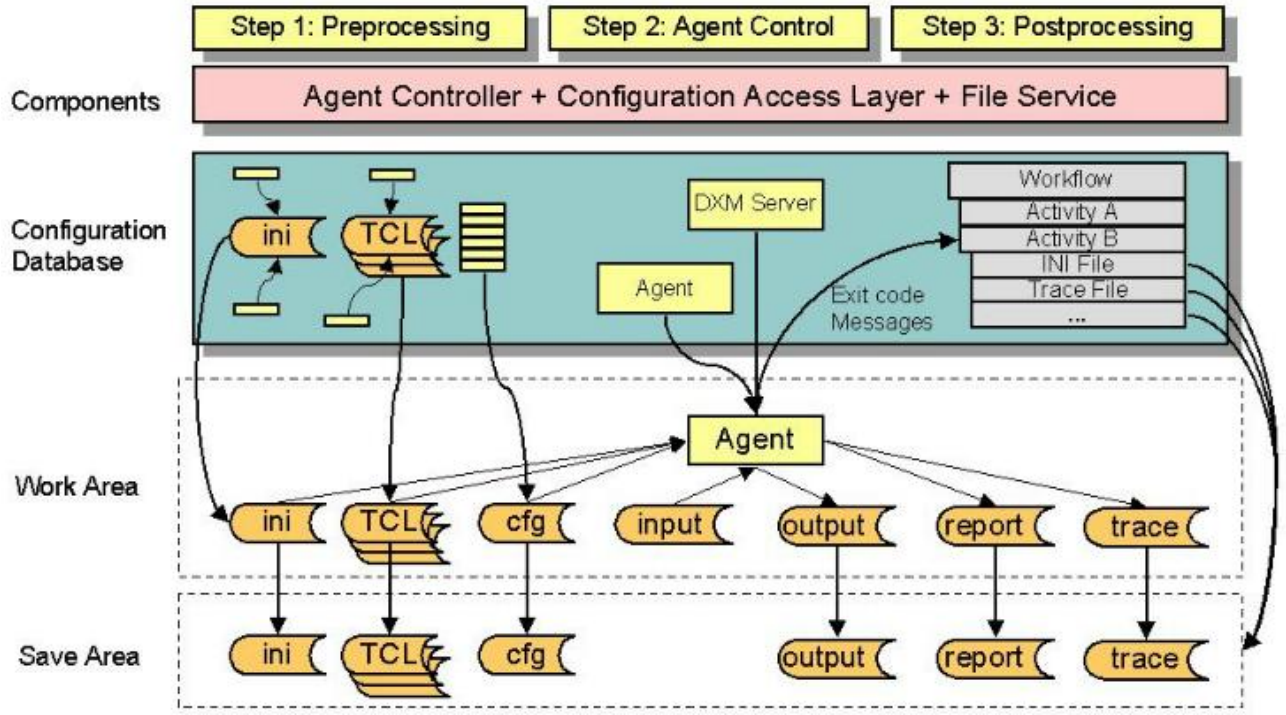



*Figure 10. Agent Controller Operations*

When errors or warnings occur during these steps, agent controller transfers the information to the status tracker, which writes it into the **Remark** field of the status entry.

The agent controller writes a "processInfo" file that contains information about internal details of this run, such as the executable that was called and the command line and the delta information that was passed back and forth.

The processInfo file contains the following fields:

- Return - the exit code that the agent returned
- Remark - messages that the agent controller generated for this agent
- DeltaOutputData - the delta information that DirX Identity extracted from the job object to deliver to the agent controller or the agent

- ExpirationTime - the defined timeout in the job definition for this agent
- DeltaInputData - the delta information that the agent has delivered to be stored in the DirX Identity database after a successful run
- CommandLine - the command line for this agent run
- Executable name - the name of the executable that was used for this run
- ProcessUserName - the user name under which this agent run was started (note: the password that was used is not displayed here)
- ProcessDomain - the domain under which this agent run was started
- JobPath - the path of the work area where this agent has been run

DirX Identity also catches any **stdout** and **stderr** output and reports it if present in the processInfo file.

#### 4.2.1.3. Workflow and Activity Execution Status Values

This section describes the status values that can occur during the run of a Tcl-based workflow or activity. These values are displayed in the DirX Identity Manager Monitor View in the **Result** field of a workflow or activity status entry.

#### 4.2.1.4. Workflow Execution Status

The following figure illustrates workflow status changes that can occur during workflow execution.

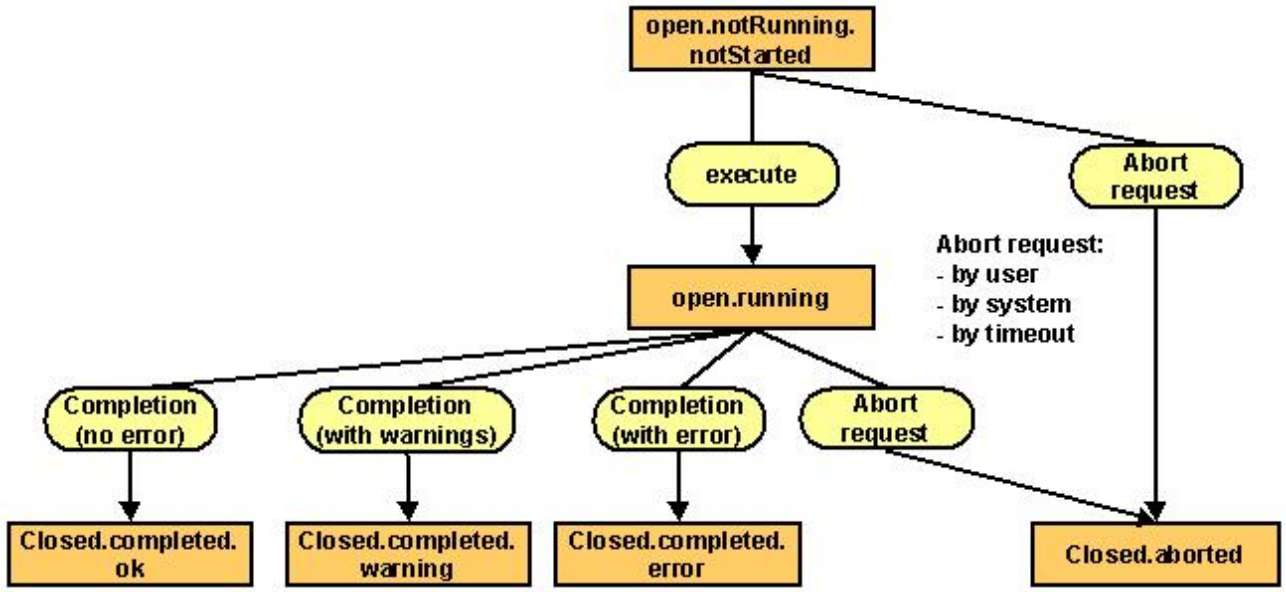



*Figure 11. Workflow Execution Status Flow*

The possible workflow execution values shown in the figure are:

- **open.notRunning.notStarted** - The workflow's initial state.
- **open.running** - The normal state of the running workflow.
- **closed.completed.ok** - All workflow activities have run successfully. This message

indicates a successful workflow run.

- **closed.completed.warning** - One ore more activities in the workflow have reported a warning. The workflow is not aborted.
- **closed.completed.error** - One activity in the workflow has not run successfully and the workflow has been aborted at this point. This message indicates an erroneous workflow run.
  Note: You can ignore this state when setting the **Ignore Errors** flag in the activity definition.
- **closed.aborted** - The workflow was aborted by a request from the DirX Identity Manager (the administrator has initiated the abort) or by the C++-based Server due to a fatal error condition or a timeout.

#### 4.2.1.5. Activity Execution Status

The following figure illustrates activity status changes that can occur during activity execution.

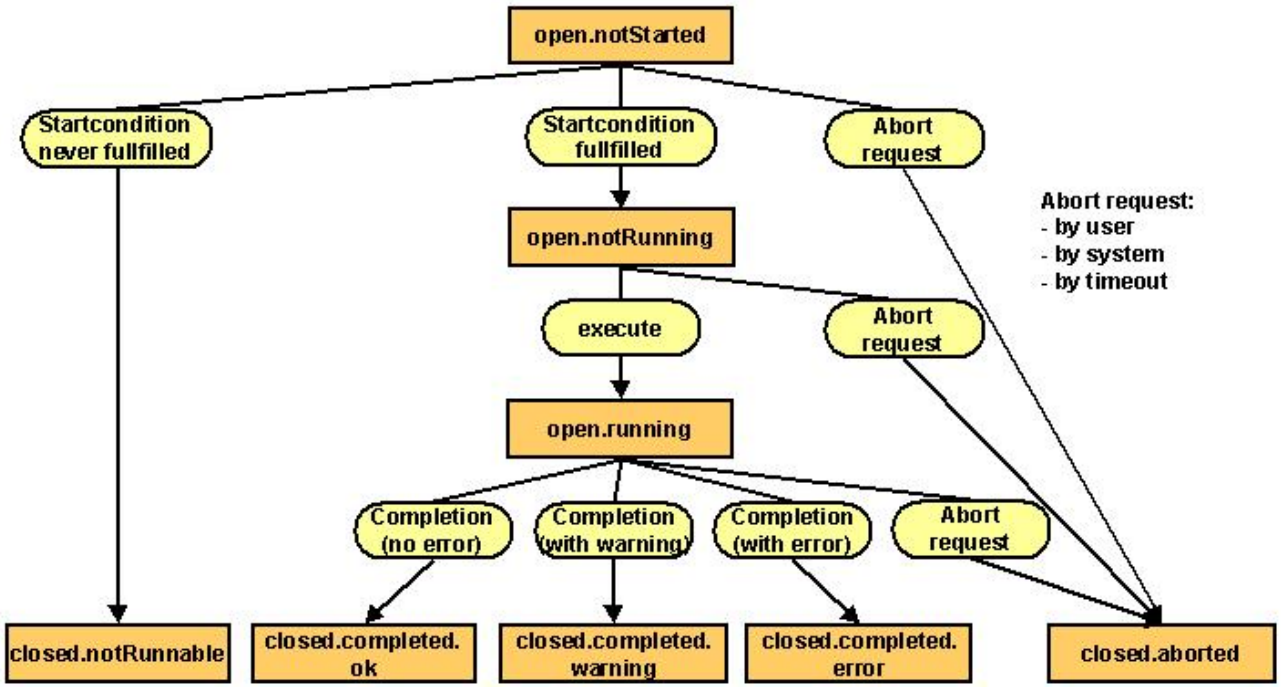



*Figure 12. Activity Execution Status Flow*

The possible activity execution values shown in the figure are:

- **open.notStarted** - The activity's initial state.
- **open.notRunning** - The start condition is satisfied, but the activity is still not running.
- **closed.notRunnable** - A start condition exists that does not allow the activity to run.
- **open.running** - The normal state of the running activity.
- **closed.completed.ok** - The agent has returned an exit code that indicates no errors or warnings which DirX Identity treats as a successful run. Please note that the incomplete generation of status entries is treated as a warning that is only reported in the remark field of the status entry. See the **OK Status** fields in the Agent and Job objects to setup

the required behavior.

- **open.completed.warning** - The agent has returned an exit code that is defined as a warning. In this case the workflow is not aborted, but the warning is reported. A warning is also reported when the agent controller detects **stdout** or **stderr** output. See the **Warning Status** fields in the Agent and Job objects to setup the required behavior.
- **closed.completed.error** - The agent has returned a non-zero exit code, which is treated as an erroneous run. The workflow will be aborted.
- **closed.aborted** - The workflow was aborted by request from the DirX Identity Manager (the administrator has initiated the abort) or by the system due to a fatal error condition or a timeout.

#### 4.2.1.6. Checkpoints

You can use the checkpoint feature for workflows that contain either the meta controller or framework-based jobs. Framework-based agents are Dashboard, SAP R3 UM and SAP EP UM.

This feature is especially useful for workflows that run for many hours. If a temporary error occurs during such a run, the workflow ends with an error. If it is re-started later on, it performs all operations from the beginning regardless of whether the job runs in delta or full mode.

If checkpointing is enabled, the relevant jobs write a checkpoint after a definable number of correctly-processed records. If the workflow runs into an error, the checkpoint value is stored via DirX Identity's delta mechanism. If the workflow is re-started, it starts the job where the error occurred and the job itself begins processing records starting from the last checkpoint value. This might result in processing some records again.

To enable the feature, set the parameters at the workflow object and/or the related parameters at the corresponding job objects. Checkpointing for the workflow and one or more jobs are independent of each other. If you only enable the workflow for checkpointing, the workflow starts with the erroneous activity on the next run. If you only enable one job for checkpointing, this job starts with the saved checkpoint of the last run. If the last job run finished without errors, no checkpoint is written and the job starts from the beginning.

The checkpointing mechanism also works with hierarchical workflows, meaning that the activity to be restarted can also be a sub workflow, whose activities can be also be configured for checkpointing.

The relevant **workflow parameters** for checkpoint control are visible in the **Operations** tab:

**Enabled**

the workflow works in checkpoint mode. One or more jobs can also be enabled for checkpointing. If no job is enabled for checkpointing, only the "Restart at Activity" mechanism becomes operative.

**Retry Limit (default is 3)**

the maximum number of retries. If the limit is reached, the workflow runs again in

complete full or delta mode (as defined). It also informs all jobs to start from the beginning even though checkpoints might have been saved. (It sets a “ignoreCheckpoint” flag in the “create” message of the activity.)

Workflow parameters that display the checkpoint status are:

**Retry Count**
: the number of retries already performed.

**Restart Activity**
: displays one or more activities that are started in parallel during a retry operation. This field is read-only.

The corresponding job parameters are visible in the **Operation** tab:

**Checkpointing Enabled**
: if this flag is set, the job writes checkpoints regularly.

**Checkpoint Frequency**
: the number of processed records after which a checkpoint is written. After a serious agent problem, this activity can start at the last checkpoint (and not from the beginning again).

**Events where Checkpoints are Saved for the Next Run**

Checkpoints are saved if the workflow completes with errors (for example, at least one activity ends with an error) or the workflow or an activity is aborted. The delta information is only saved in each job configuration entry if the workflow ends without errors.

If both delta and checkpointing are enabled for one job, the delta info item containing either delta or checkpoint information is saved at the end of the workflow regardless of whether the workflow ended with or without error.

#### 4.2.1.7. File Paths and Areas

The DirX Identity runtime environment requires three file paths for each C++-based Server to work properly:

- Installation path - the path to the installation area where DirX Identity is installed. For example: C:\Program Files\Atos\DirX Identity

  Do not change this field because it is handled automatically by the installation routine.
- Work path - the path to a work area that stores all files created during the execution of an activity when relative paths are used. For example: C:\Program Files\Atos\DirX Identity\work
- Status path - the path to a status area that stores all permanent files after the execution of an activity for each workflow run. For example: C:\Program Files\Atos\DirX Identity\status

You can set up the work and status paths to conform to your requirements. You can also

distribute both areas on different disks to ensure that full status areas do not disturb the runtime environment.



When the status area is inaccessible or full, the C++-based Server does not stop further workflow runs and does not regard this as erroneous result of a workflow run. However, it logs and reports the problem.

**Work Area**

The structure of this storage area is:

**work***workflowFolderStructure*_workflow__activity_

Where *workflowFolderStructure* is equal to the structure of the workflow folder in the Connectivity configuration under the object Workflows, for example for the workflow

Workflows\Default\LDIFfile\LDIFfile2Meta

the work area path is

...work\Default\LDIFfile\LDIfile2Meta

Below this path additional activity folders are created. In this case:

...work\Default\LDIFfile\LDIfile2Meta\LDIFfile2Meta_MetaCP

This file directory contains all input, output, delta, configuration, report and trace files of a specific workflow and activity (DirX Identity uses the display names of the objects) that are flagged with **Save mode** = PERMANENT. For example:

...work\Default\LDIFfile\LDIfile2Meta\LDIFfile2Meta_MetaCP\control.tcl

**Status Area**

The structure of this storage area is:

**status***workflow.starttime*_activity.starttime_

For example for the workflow

Workflows\Default\LDIFfile\LDIFfile2Meta

the status area path for the workflow is

...status\LDIfile2Meta.200301131420l1Z

Below this path additional activity folders are created. In this case:

...LDIfile2Meta.200301131420l1Z\LDIFfile2Meta_MetaCP.20030113145822

This activity folder contains all copied input, output, delta, configuration, report and trace files from a specific agent run (the files that are flagged with **Copy to status area**. For

example:

...LDIFfile2Meta_MetaCP.20030113145822\control.tcl

### 4.2.2. Configuration Files and Tcl Scripts

DirX Identity Connectivity stores all of the configuration information that an agent requires in attributes (object properties) within the configuration. Agents typically require this configuration information to be made available to them through files. The DirX Identity agent programs (which use "*.ini" configuration files) and the meta controller (which is controlled by Tcl scripts) read their operating instructions from these files, which must exist prior to their execution. The meta controller also requires attribute configuration files that describe the input and output format information.

For example, for import to a DirX Identity store via the meta controller, the agent configuration information is contained in Tcl scripts. For import into an Active Directory connected directory, the agent configuration information is contained in an "ini" file.

The meta controller Tcl scripts (called the synchronization profile) are generally separated into a well-defined structure. The most important parts are:

- The control script, which is called from the command line. The control script contains a list of variables that define all of the necessary parameters for connecting to directories or opening files in read or read/write mode, the attribute configuration and selected attributes for both the source and target directory, the export and import parameters, and so on. The control script calls the profile script, which in turn calls the mapping procedure.
- The profile script, which contains the main code of the synchronization procedure. Its structure can vary dependent on the synchronization scenario and it is normally based on a lot of standard code delivered with DirX Identity.
- The mapping script, which contains the entire code for performing an attribute mapping according to the settings made at the user interface level, for example by filling a table with appropriate assignments of target attributes via mapping procedures to source attributes.



Central use of mapping scripts is only possible for Tcl Mapping Scripts. Table based mapping scripts must be located directly under the job object and cannot be centralized!

For details, see the section "Tcl-based Connectivity Standard Script" in the *DirX Identity Application Development Guide*. The pre-configured objects contained in the default applications delivered with DirX Identity use this Tcl script structure.

The meta controller needs attribute configuration files in addition to the Tcl scripts. Attribute configuration files describe the static content of the connected directories, such as the objects and attributes of a database (for LDAP, ODBC or other formats) or the format and content of files (for tagged files like LDIF or non-tagged files like CSV and XML). See the *DirX Identity Meta Controller Reference* for further details.

A DirX Identity Connectivity job can handle:

- One ini file
- One control Tcl script
- One profile Tcl script
- One mapping Tcl script
- An unrestricted number of additional Tcl scripts

A DirX Identity connected directory provides one attribute configuration object, which is used by the meta controller to generate the corresponding attribute configuration file.

The number and kind of configuration files depends on the type of agent. The generic agent type, which is the basic agent type that DirX Identity provides (and from which all other agents are derived), contains all of these files. Consequently, the number of configuration files that DirX Identity can handle is not restricted.

The agent configuration files contain a lot of information that already exists in the properties of configuration objects in the configuration database. For example, the "server" parameter in an agent configuration file can map to the TCP/IP address of a particular service object.

You can set references to these object properties in the agent configuration file to guarantee consistency between the parameter in the configuration file and the setting for the property in the configuration database.

At each runtime, DirX Identity regenerates the agent configuration file from the property information in the configuration database. At this time, it resolves the references contained in the configuration file and updates the file with any configuration changes made since the last run. Thus configuration information is always up-to-date.

## 4.2.3. Object Naming

DirX Identity supports two object naming mechanisms:

- Object identifiers
- Display names

#### 4.2.3.1. Object Identifiers (common names)

When administrators work with several instances of DirX Identity Manager in parallel on the same configuration database, it is possible that the two different Manager instances can create the same object identifiers (the common names in the LDAP directory) at the same time because both administrators choose the same name. To avoid this problem, the DirX Identity Manager creates unique object identifiers (IDs). These IDs are difficult for users to read and remember (you can see them in the data view of the DirX Identity Manager when looking at newly created configuration objects in the Connectivity configuration tree). For example:

```
uid-c0671bf6:1ac93:e734b95f23:-7ff9
```

You can define a readable display name for each configuration object, which is stored in the attribute dxmDisplayName. You can rename this display name as necessary without disturbing the internal structure of the configuration database, because the ID is used for all references (with the exceptions described in the topic "Display Names").

If you delete an object and create it again with the same display name, it has a different ID but looks to be the same. References contained in the configuration database that pointed to the old object **will not** point to the new object. You must set all references manually again to the new object. Use the **Show References** feature to find all incoming references.

You may notice that a lot of objects in the Connectivity configuration have readable common names; for example, BA_ADS for the connected directory that has the display name ADS. This readable naming results from the creation method used in the beginning.

#### 4.2.3.2. Display Names

DirX Identity Manager administers unique display names in a specific subtree of the configuration database because it would be very confusing to see two different objects with the same name at the user interface level.

However, there are circumstances in which display names are used outside of the configuration database:

- DirX Identity uses the activity and workflow display names as part of the work and status area paths. If your activities keep any files that affect the next run of the workflow-for example, a delta database-and you change the display names of either of these two objects after you have run the workflow for the first time, DirX Identity relocates the work path to the new display name(s) and the delta database remains in the old location. As a result, your workflow will perform a full operation instead of a delta run. Currently DirX Identity does not warn you if you rename an activity or workflow name. You must know if there will be a problem (if the workflow has never been executed you will not have a problem) or you must solve the problem (for example, by moving the delta database to the new work path).
- The DirX Identity reference mechanism (see the chapter "Customizing Object References" in the *DirX Identity Customization Guide*) is built so that references are as stable as possible and remain independent from values in the configuration database. Sometimes you must use other attributes in references to specify a particular value of a multi-valued attribute, for example, for distinguished names (DNs). Do not use the display name because the reference is broken when you change the display name or when it is set automatically during a copy operation because the reference is not updated automatically. DirX Identity detects the break during the next workflow run, resulting in an error. DirX Identity cannot detect the effect on reference definitions in configuration files in advance because this action would be extremely time-consuming. You should avoid this behavior in many cases by using the dxmAnchor attribute, which is not changed during copy operations in contrast to the display name.

### 4.2.4. Linking Objects

DirX Identity supports different mechanisms to use objects; for example, attribute configuration or Tcl files. You can locate objects either locally under the parent object or in

a central place.

Objects are used by setting a link from the parent object to the required object. Depending on the location, the result will be different:

- When you reference a central object, changes to this object will affect all parent objects and their workflows. This behavior is nice when you need a central place to change parameters. On the other hand, changes to the object (especially when performed with a wizard, where you don't see the object structure) may affect workflows that you do not intend to change. DirX Identity displays a **red border** with the text "This object might be shared since it belongs to the Configuration Folder" to indicate the use of central objects.
- Referencing a local object allows you to set all parameters individually. However, when you need to change a central parameter (which is contained in several local objects), you must make the change multiple times, and you may miss making the change to one of the objects
- Local objects can also be used by several parent objects, which creates a type of quasi-central object (shared objects). The same problem can arise as with central objects regarding the change of attributes. In addition, you may unintentionally delete an object that is still used by another one. Use the option "Check references to avoid broken links" when deleting objects to avoid such situations.

## 4.2.5. Understanding Notifications

DirX Identity provides two types of notifications in the Tcl-based environment:

- Change notifications - the meta controller calculates all necessary actions and changes to update the directory. Optionally it can send a JMS message that contains all change information. You can set up Java-based workflows to interpret and process these messages.
- E-mail notifications - if program logic cannot solve outstanding issues or for informational purposes, you can produce e-mail notifications.

#### 4.2.5.1. Change Notifications

Before performing updates at the directory server, the meta controller calculates all necessary actions and changes. It can optionally send this information in the form of a JMS message for further processing. Set the relevant change notification parameters to enable this feature.

#### 4.2.5.2. E-mail Notifications

DirX Identity provides a notification agent, which is a separate executable that can be used to send an e-mail message from within a Tcl script using the **metacp exec** operation. You control the notification agent through a simple command sequence and a configuration file in XML format that defines all parts of the message. As with Tcl or "ini" files, the agent controller writes the notification XML configuration file to the work area. All the operations for files (for example, deleting files in the work area and moving them to the status area) are also available for notification files.

You can send a notification from within a Tcl script with the following command sequence:

```
proc calculateJavaPath { } {
    global env
    regsub -all "\\\\" $env(DIRXMETAHUB_INST_PATH) "/" install_path
    source "$install_path/basic.input.tcl"
    return $DXI_JAVA_HOME
}
regsub -all "\\\\" $env(DIRXMETAHUB_INST_PATH) "/" install_path
set java_dir [calculateJavaPath]
if {$tcl_platform(platform) == "windows"} then {
    set cmd "exec \"$java_dir/bin/java\" -Duser.language=en -cp
\"$install_path/lib/java/ext/javax.activation.jar;$install_path/lib/j
ava/ext/mail.jar;$install_path/lib/java/notify.jar;$install_path/lib/
java/dxcCrypto.jar;$install_path/lib/java/dxmUtil.jar"
siemens.dxm.notify.sendmail \"$notify_cfg_file\" "
} else {
    set cmd "exec \"$java_dir/bin/java\" -Duser.language=en -cp
\"$install_path/lib/java/ext/javax.activation.jar:$install_path/lib/j
ava/ext/mail.jar:$install_path/lib/java/notify.jar:$install_path/lib/
java/dxcCrypto.jar:$install_path/lib/java/dxmUtil.jar"
siemens.dxm.notify.sendmail \"$notify_cfg_file\" "
}
```

In this sequence, DirX Identity evaluates the fully qualified pathname first. It then calls the notification agent; the calling syntax depends on the platform on which the Tcl script is executing. The procedure **calculateJavaPath** computes a representation of *dxi_java_home* as needed in Tcl scripts.

The notification agent uses the XML configuration file to process the request. It processes an e-mail notification, which must contain addresses (From, To, CC (optional)), a subject, a message text (optional and with references to be resolved) and attachments (optional; for example, a log or data file). Note that you can configure only one **To** address because DirX Identity assumes that there is exactly one entity responsible for the requested action. You can define as many **CC** addresses as you like.

The notification agent returns the following exit codes:

- **Exit code = 0:** The message has been sent correctly.
- **Exit code = 50:** General error.
- **Exit code = 51:** The sending of this message has failed.
- **Exit code = 52:** The command line for the notification agent contains an error.

Example: java siemens.dxm.notify.sendmail .\config.xml

- **Exit code = 53:** The type is not **email**.
- **Exit code = 54:** The XML description contains a syntax error.
- **Exit code = 55:** One of the following entries is missing:
  - Type
  - Host
  - From
  - To
  - Subject

The Tcl procedure should handle these exit codes and set an exit code that the agent controller can process.

You can define any type of notification that can be performed from within a Tcl script. For example, you can define notifications that send an e-mail with the trace file when a specific error condition occurs during a workflow run.

Most of the workflows provided with DirX Identity use two standard notifications: **Notify if not OK** and data notification (**NotifyData**). The data files for these notifications are centrally configured in the **Configuration → Standard Files** folder for ease of administration. You use the job's **Notification** and **Operation** tabs to control how these notifications are used during the **metacp** job's operation. You can also create customized notifications using the Notification configuration object. The next sections provide more information about these features.

#### 4.2.5.3. Notify if not OK

The "notify if not OK" notification sends a message if the workflow runs on a warning or error condition. This notification is available for all meta controller jobs and is configurable via the **Notify if not OK** field in **Notification Control** in the **Operations** tab in the job object. Possible values are:

- **0** - no notification at all
- **1** - send notification if warnings occur
- **2** - send notification if errors occur
- **3** - send notification if warnings or errors occur

The **Notification** tab in the job object also manages notification objects. Click this tab to define all notification parameters.

#### 4.2.5.4. Data Notification

The data notification feature sends data change information. In the **Entry Handling** tab of input or output channels, you can define whether the processed data changes are:

- Automatically processed by the relevant agent, or
- Sent via e-mail to an administrator, who then handles the event by hand.

You can also choose automatic processing and in-parallel notification to the administrator.

The data notification feature is controlled by the three drop-down lists in the **Entry Handling** tabs of input or output channels. The fields are:

- **Add Entries**- controls the behavior for add operations:
  NONE - no add operation at all
  ADD - Adds are performed only automatically
  NTF - Adds are performed by notification (no automatic processing)
  ADDNTF - Adds are performed automatically and notification is sent
- **Modify Entries** - controls the behavior for modify operations:
  NONE - no modify operation at all
  MOD - Modify operations are performed only automatically
  NTF - Modify operations are performed by notification (no automatic processing)
  MODNTF - Modify operations are performed automatically and notification is sent
- **Delete Entries** - controls the behavior for delete operations:
  NONE - no delete operation at all
  DEL - Deletes are performed only automatically
  NTF - Deletes are performed by notification (no automatic processing)
  DELNTF - Deletes are performed automatically and notification is sent

In a possible scenario, the administrator handles add and delete operations by hand and modifications are performed automatically.

#### 4.2.5.5. Creating Your Own Notifications

You can extend workflows to perform your own notifications. To extend the workflows:

- Create a notification object under the relevant job object (this is currently only possible for meta controller jobs) or copy an existing one from a job object (**Copy Object**) and move it to the target job object (**Move Object**). You can also set up a central notification object in the **Configuration → Notifications** folder.
- Set all parameters (for example, the From and To fields) and adjust the text in the XML file (tab **Content**) to your needs.
- Define all files that are to be sent during the notification as attachments.
- Select the **Operation** tab in your job object and set the link to the newly created notification object.
- Call the notification from your Tcl scripts (use, for example, the user hooks **uh::Epilog**).

### 4.2.6. Workflow Design Rules

When you create or run Tcl-based workflows, you should follow some basic rules. The next sections provide information on:

- Rules for algorithms

- Rules to set up correct schedules to avoid timing conflicts
- Rules for backup and restore

#### 4.2.6.1. Rules for Algorithms

When you design your synchronizations, you must be sure not to produce conflicts. This section provides some rules to avoid problems in this area.

##### 4.2.6.1.1. Single Master System Design

If you follow the rules in this topic, your system will be very robust. Conflicts cannot occur by design.

- Define only one master for entry creation and deletion.
- Run only workflows with non-overlapping execution times. If this is not possible, be sure that the workflows do not influence each other.
- If workflows create or use the same files (for example, one workflow creates an LDIF file, the other reads it), be sure that the execution times do not overlap.

##### 4.2.6.1.2. Multi Master System Design

The rules in this topic help to control a multi-master system and minimize conflicts.

- If you need to set up several masters for entry creation and deletion, define unique criteria to distinguish the corresponding entries. Two examples are:
- Create different trees in your directory that belong to the different masters.
- Set up attributes or object classes that allow for distinguishing between the entries of the different masters, for example by filtering expressions.
- If you cannot follow the previous guideline, be sure that workflows that create or delete entries do not run at the same time (avoid overlapping execution time - see the next topic).
- If workflows create or use the same files (for example one workflow creates an LDIF file, the other reads it), be sure that the execution times do not overlap.

#### 4.2.6.2. Rules for Schedules

Many issues must be considered when scheduling workflows. This topic explains the rules and their reasons. Set up schedules so that:

- They do not have unrealistic values; for example, an interval of 0, which means that the workflow would always run - the scheduler ignores schedules with this setting.
- Their possible start times cannot overlap
- Their execution times cannot overlap

DirX Identity has some built-in features to handle conflicting situations, but only when the parameters are set correctly. For example, the DirX Identity default applications use the dxmOprMaster attribute to distinguish he mastership of different source connected

directories.

#### 4.2.6.2.1. Starting the Same Workflow Twice

The C++-based Server does not allow starting the same workflow twice (for example from the scheduler and from the DirX Identity Manager by hand). Both workflows would try to open, read and write the same files in the work area, which cannot work properly.

A message is written into the server logging and an extra status entry is written for the second instance.

#### 4.2.6.2.2. Starting Workflows after the Server Starts

At the time when the C++-based Server is started, all workflows are started where the start time has passed and the deviation time is not over.

In this case, it can happen that a workflow was previously aborted (during the stop sequence of the C++-based Server). The messaging service keeps the related start message and resends it after C++-based server startup. The scheduler at the same time tries to start the same workflow. This operation will fail, because the same workflow was already started (a message is written into the server log and a status entry that indicates a double workflow start is created). In this way, DirX Identity ensures that the workflow is only started once.

#### 4.2.6.2.3. Basic Rules for Central Configuration Object Parameters

This section describes the basic rules for setting the values of parameters in the central configuration object.

#### 4.2.6.2.4. Rule 1: Polling time

- The **Polling Time** in the central configuration object **must be greater than 0**. Otherwise, the scheduler will assume a default of 5 seconds.

  Reason: a timer based on the polling time activates the scheduler. A polling time of 0 would result in heavy system load. Therefore we do not permit you to set this value.

#### 4.2.6.2.5. Rule 2: Time Interval

- The **Time Interval** in the central configuration object **must be greater than 0**. Otherwise, the scheduler will assume a default of 1 day (86400 seconds).

  Reason: this value determines the interval at which the status tracker removes status entries from the database. A Time Interval of 0 would result in heavy system load. Therefore, we do not permit you to set this value.

#### 4.2.6.2.6. Rule 3: Schedule Sync Interval

- The **Schedule Sync Interval** of the central configuration object **must be greater than 0**. Otherwise, a default of 1 hour (3600 seconds) will be assumed.

  Reason: normally the DirX Identity Manager automatically informs the scheduler when schedule information is changed. If this operation fails, this mechanism reads the

schedules regularly from the configuration database. A Schedule Sync Interval of 0 would result in heavy system load (the schedule reread would happen always). Therefore we do not permit you to set this value.

#### 4.2.6.2.7. Basic Rules for Schedule Object Parameters

This section describes the basic rules for setting the values of parameters in the schedule object.

#### 4.2.6.2.8. Rule 4: Time Interval

The **Time Interval** in a schedule object must be

- **Greater than 0**.

  Reason: this value determines the interval after which this workflow has to be started again. A Time Interval of 0 does not make sense. Therefore we do not permit you to set this value.
- **Greater than the polling time** (see central configuration object). The scheduler will ignore schedules that do not satisfy this condition. Related error messages will be written into the logs (event log or log files).

  Reason: A time interval that is smaller than the polling time could result in missed starts of workflows.
- **Greater than the sum of deviation + timeout + 5 * polling time**. The **time buffer** of 5 * **polling time** is recommended because multiple server components (scheduler, workflow engine, agent controller, messaging service) are involved in aborting a workflow when timeout is exceeded. Here, timeout is the sum of all job timeouts of the related workflow, multiplied with (1 + latency-factor / 100).

  Example: Suppose that a workflow consists of 2 jobs running in a sequence, both with a job timeout of 10 minutes, and the latency factor is 20. Then timeout (of the workflow) is 24 minutes.

  Reason: The schedule might miss start times when this condition is not satisfied.

#### 4.2.6.2.9. Rule 5: Deviation

- The **Deviation** of the schedule **must be greater than the polling time** (see central configuration object).
  Reason: When this condition is not satisfied, the scheduler can miss workflow start times.

#### 4.2.6.2.10. Combined Rules

These rules must be considered when several workflows that could influence each other shall not be started simultaneously or may not have overlapping execution times.

To give full control to the C++-based Server (for example, to abort agents after the timeout has been reached), you must set the **Abort Execution allowed** flag in the agent object in

the Expert View.

#### 4.2.6.2.11. Rule 6: Non overlapping start times for workflows

Suppose you don’t want the scheduler to start a set of workflows simultaneously (we assume that the workflows' runtime is no longer than 30 minutes). Define the related schedule’s start time, deviation and interval parameters so that the intervals of possible start times do not overlap. The following table provides an example of non-overlapping start times.

*Example 1. Non-overlapping Start Times*

| Schedule | Start Time | Interval | Deviation |
|---|---|---|---|
| A | 01.01.2001 00:00:00 | 2 h | 30 minutes |
| B | 01.01.2001 01:15:00 | 2 h | 30 minutes |

The scheduler will not start the related workflows simultaneously, as the following figure illustrates.

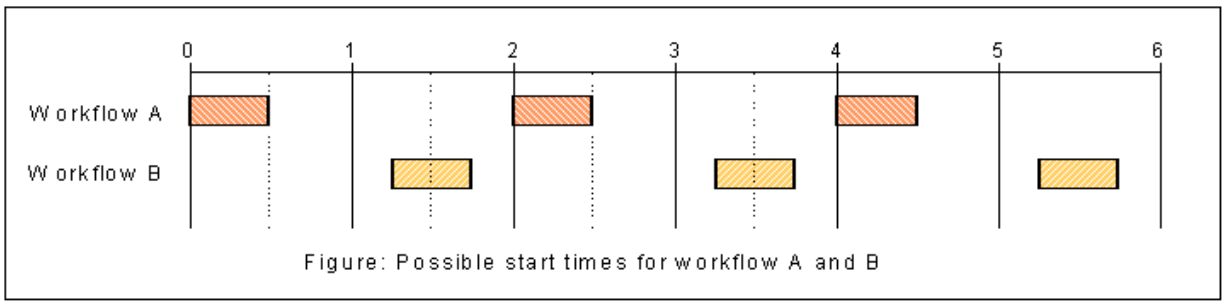



*Figure 13. Non-overlapping Start Times: an Example*

*Example 2. Overlapping Start Times*

If we change the interval of workflow B to 3 hours, then overlapping starts would be possible at 4:15 to 4:30, 10:15 to 10:30 and so on.

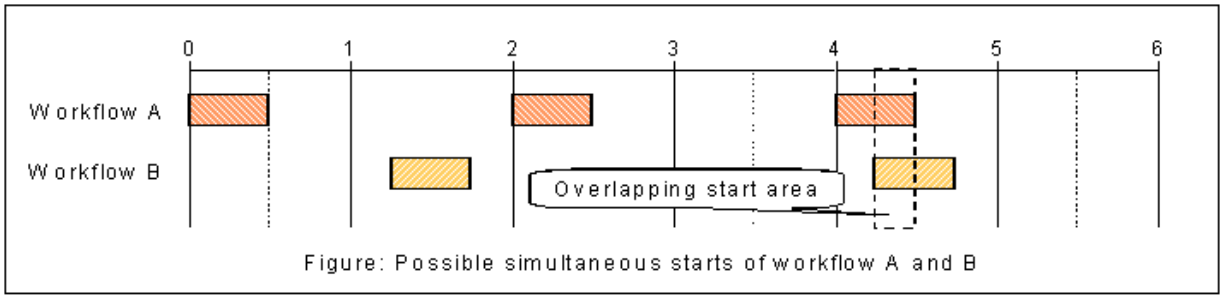



*Figure 14. Overlapping Start Times: an Example*

We must also take into account that after a restart of the C++-based Server all workflows that are in their deviation time will be started simultaneously at the same time (the related messages are waiting in the messaging service). Let’s assume that the server is restarted at 4:20; then both workflows are started immediately.

#### 4.2.6.2.12. Rule 7: Non overlapping execution of workflows

Suppose you want to schedule workflows so that their execution does not overlap (which may be convenient for resource-consuming workflows). Then you must define the related schedule parameters **start time**, **deviation, interval** and **polling time** so that the intervals of possible execution times of a workflow do not overlap.

*Example 3. Non overlapping Execution Times*

| Schedule | Start Time | Interval | Deviation | Timeout |
|---|---|---|---|---|
| A | 01.01.2001 00:00:00 | 2 h | 15 minutes | 30 minutes |
| B | 01.01.2001 01:00:00 | 2 h | 15 minutes | 30 minutes |

We additionally suppose that the polling time is set to 5 seconds.

As shown in the next figure, the workflow execution cannot overlap.

*Non-overlapping Execution: an Example*

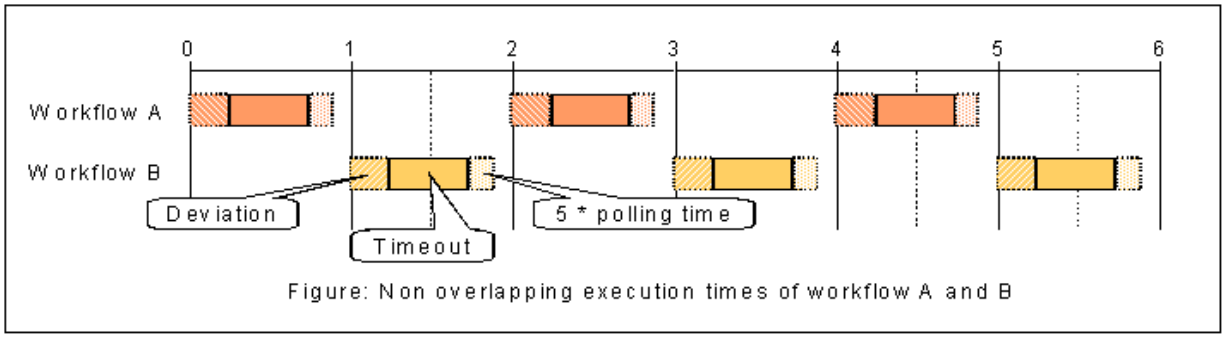


Figure: Non overlapping execution times of workflow A and B

In normal operation, the upper limit of a life time of a workflow instance in the C++-based Server will be **timeout + polling time**, because there may be a delay of polling time before scheduler and workflow engine really detect a timeout.

The example shows that a workflow can only be active in time intervals between **start time *n*** and **start time *n* + deviation + timeout + 5 * polling time**, where start time *n* = start time + *n**interval (*n* being an integer).

### 4.2.6.3. Rules for Backup and Restore

If you want to back up and restore your LDAP server or the configuration part of it, we recommend the following procedure:

- Deactivate all schedules to ensure that no new workflows are started (use the Disable Scheduling function).
- Check with the DirX Identity Manager's Get Server State function on each C++-based Server in your environment that no workflows are still running (only status tracker and scheduler threads may be visible in the details view of this feature).
- Stop all C++-based Servers.
- Perform a backup or restore of your server with the native methods of your LDAP

directory (for DirX, you will normally use DirXmanage) or perform an Export Configuration or Import Data command from the DirX Identity Manager's Expert View.

- After a restore, restart all DirX Identity Managers (because the content of the configuration database could be changed completely and the caches must be cleared).
- Restart all C++-based Servers.
- Activate all schedules (use the Enable Scheduling function).

Normal operation is now restored.

After a restore of a complete database or only the configuration tree in the database, you must be aware of the following problems:

- If you have restored an older version of your configuration tree, status entries in the Monitor View of the DirX Identity Manager may be missing. Consequently, you can no longer access the related status entry file information in your status area in the file system.
- If your database comes from a different environment (for example from a test environment), some settings in your database may not be correct, and you must adjust them by hand. Candidates are the settings in the service objects and others that are different in the two environments. Consequently, we do not recommend exchanging whole databases or configuration trees. Use instead the Export data / Export Subtree and Import data feature of the DirX Identity Manager to exchange parts of the configuration database (a whole scenario, workflows, jobs or connected directory objects).
- Do not try to use the Import data method with data that was created with a previous version of DirXmetahub V6 if this was not explicitly allowed (see the release notes). Changes in the data structure may no longer fit with the new version of DirX Identity.

## 4.2.7. Copying Tcl-based Provisioning Workflows

You can copy Tcl-based Provisioning workflows either in DirX Identity Manager's Global View or in its Expert View. The next sections describe how to perform these tasks. For details about DirX Identity Manager, see the chapter "Using DirX Identity Manager" in the *DirX Identity User Interfaces Guide*.

### 4.2.7.1. Copying Workflows in the Global View

To make copy operations easier and more intuitive, DirX Identity provides a complex copy method that is activated within DirX Identity Manager using the **Configure** method for connected directories and the **New** method for workflows. Both methods work closely together.

First, you create new connected directory icons that represent your connected directories. With the **Configure** method, you can configure these objects based on other existing connected directories that act as templates. This template is copied to represent your new connected directory. Note that the related service object is also copied, which avoids interference with the original object.



To avoid copying files and channels that will never be used, the copy operation does not copy files and channels of connected directories of type **File**. Instead, the data files and channels are created when a workflow is copied. As a result, it does not make sense to create data files during such a copy operation.

When two or more of these connected directories exist, you can link these objects via workflow lines. Next, you can either link existing workflows to these lines (using the **Assign** method) or create copies of existing workflows (using the **New** method) that act as templates. This template is copied to represent your new workflow.

Copying these templates is a complex procedure that can handle most variants of workflows:

- By default, DirX Identity tries to find all workflows that fit between the two connected directories. It checks the **Type** field of the connected directory type objects of the connected directories at both sides of the workflow line and compares them with the **Endpoints** field of all workflows. Matching workflows are displayed in the template chooser dialog. If no workflows can be found, an error message is displayed.
- In some cases, the list does not contain the workflow you want to copy. Deselect the **Matching endpoints** flag to show the list of all workflows. Now you can freely choose any of the workflows. Using this flag is especially useful if you intend to create a new type or variant of workflow starting with an existing one.
  Note: The copy procedure may run into an error if the workflow requires resources that are not available between the two connected directories. In this case, you need to copy and change the workflow in the Expert View.
- If the connected directories are of the same type, you are asked about the direction the workflow is to work (for example, for an LDAP to LDAP workflow).
- The workflow is copied, including all activities.
- All jobs referenced by these activities including all sub-objects are copied, too. If an activity references a workflow, the workflow is not copied (a warning message is shown).
- The input channel of the first activity (the start activity) is mapped to the related new connected directory. This action creates a new channel that points to the source connected directory. If the connected directory type is **File**, the necessary data file to which the channel points is also created.
- The output channel of the last activity (end activity) is mapped to the related new connected directory. This action creates a new channel that points to the target connected directory. If the connected directory type is **File**, the necessary data file to which the channel points is also created.
- When copying nested workflows, you can choose with the **Copy workflows recursively** flag whether the sub workflows shall be copied or not. If you do not copy the sub workflows, the copied workflow points to the original workflows. You can change these links later on by hand.

Some types of workflows cannot be copied completely with the **New** method and thus must be copied partially with other methods:

- Nested workflows can only be copied when the source and target connected directories of the sub workflows are the same as for the top-level workflow. You must copy workflows that do not meet this restriction by hand.
- For workflows that contain more than one start or end activity, DirX Identity tries to find a connected directory that fits with the corresponding connection of the workflow line and maps the new connected directory with the original one. Connected directories that do not map are not re-mapped; you must take care of this step by hand afterwards.

Both the **Configure** method for connected directories and the **New** method for workflows keep the original folder sub structure in the scenario to which the objects are copied. You can change these structures at any time with the **Move** method in the Expert view if required.

#### 4.2.7.2. Copying Workflows in the Expert View

If you copy an object in the Expert View, the complete subtree is copied. For example, copying a workflow copies the workflow object and all activities with all links to other objects. Linked objects (like jobs or channels) are not copied. As a result, you must copy the linked objects separately and change the links accordingly.

For example, if you copy workflow W1 with activity A1 that points to a job J1, this results in workflow W2 with activity A1 that points to job J1. Note that the activity name can stay the same because it resides under a different workflow object.

In this example, you must copy job J1 separately, which results in job J2 (with all its sub-objects like configuration files, local connected directories or trace files). Now you must re-link activity A1 from workflow W2 to job J2. Now the two workflows (W1 and W2) are independent down to the job level.

Because job J2 still points to the same channels as job J1, you must repeat the procedure for the channels and eventually the connected directories.

### 4.2.8. Starting Tcl-based Provisioning Workflows

You can start Tcl-based Provisioning workflows in several different ways:

- To run a workflow only from time to time or to test workflows, you should use the **Run** option of the workflow line context menu in the DirX Identity Manager Global View or the **Run Workflow** or **Run Activity** options of the context menus of the workflow or activity objects in the DirX Identity Manager Expert View.
- You can use schedules to run workflows regularly at well-defined times.
- You can use the **runwf** tool to trigger a workflow start from any event. See the chapter "Using DirX Identity Utilities" in the *DirX Identity User Interfaces Guide* for more information.

If you try to start a workflow that is still running, DirX Identity uses the following procedure to handle this situation:

- When a workflow is started and a previous instance is still running, the workflow request is buffered as a waiting thread.

- When the previous workflow ends it starts the next workflow (the next waiting thread) automatically.

Because the number of threads in DirX Identity is limited (see the **Max number of threads** parameter of the C++-based Server), the buffering stops when two-thirds of this number is reached and a "Second workflows instance" message is sent. The rest of the threads are left for other workflows to be run. Otherwise one client could request many workflow starts that fill up the server and prevent any other workflow from starting.

Raise the Max number of threads parameter if you need a higher number of buffered workflows.

When the Max number of threads is reached, an error message is sent as in previous versions of DirX Identity.



These waiting threads are visible in the Process table of the Monitor View (entries without a time extension of the display name) as well as in the Get Server State table.

# 5. Managing Passwords

Password management in Connectivity means managing password synchronization, including:

- Setting up and maintaining the scenario structure
- Setting up and maintaining the password workflows

The sections in this chapter describe:

- How DirX Identity's password management feature works (generic operation)
- How password changes flow from Windows domains
- How password changes flow from Web applications
- How to configure the password synchronization workflows
- How the password synchronization workflows operate
- How the password change algorithms operate

For troubleshooting hints in a password management scenario see the section "Password Synchronization" in the *DirX Identity Troubleshooting Guide*.

## 5.1. Understanding Password Management

DirX Identity password management comprises a set of comprehensive features to set up, maintain and distribute user passwords in a distributed environment:

- The DirX Identity Web Center allows for user password management via self-service and administrative applications.See the *DirX Identity Provisioning Administration Guide* for more information.
- The DirX Identity Windows Password Listener captures passwords at Microsoft Windows domain controllers.
- The DirX Identity Password Event Manager receives password change notifications and processes them.
- The DirX Identity Password Change workflows transfer the changed passwords via connectors immediately to the connected target systems.

The following figure shows how these components interact:

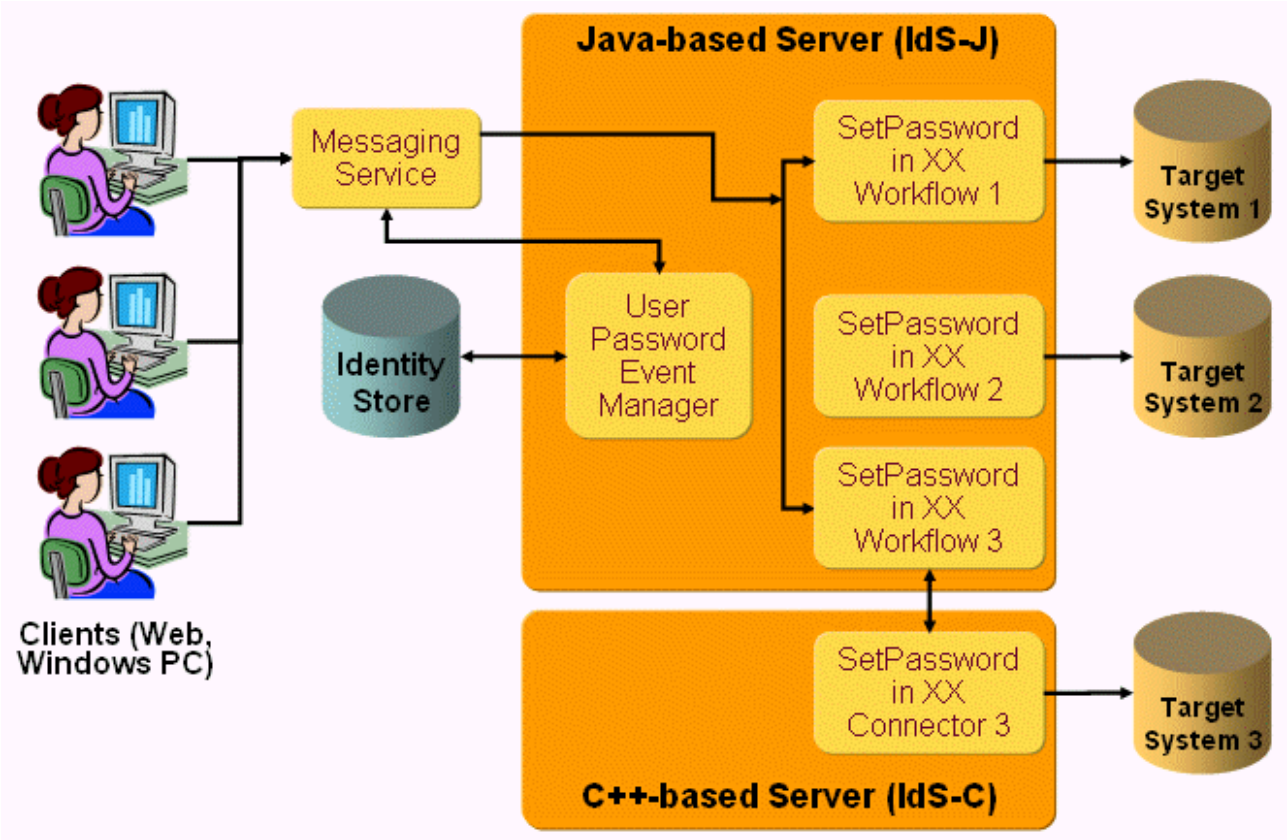



*Figure 15. DirX Identity Password Management Component Interaction*

As shown in the figure:

- A user changes his password via the Web Center (see the *DirX Identity Provisioning Administration Guide* for more information).The Web Center sends the password change notification to the DirX Identity Messaging Service.
- Alternatively a user can change his password in the Windows domain.The Windows Password Listener captures this password change and sends it to the Messaging Service (see the *DirX Identity Connectivity Reference* for more information).
- The User Password Event Manager subscribes to these notifications, joins the event to the correct user in the Identity Store, changes the password for the user and issues change events for all accounts in all target systems where this change is required.
- The User Password Event Manager subscribes to these notifications, joins the event to the correct user in the Identity Store and processes change events for all accounts in all target systems where this change is required.
- Target system-specific Set Password in *XX* workflows subscribe to these change events and perform the necessary password change via the related connector in the connected system.To support reporting and real-time status display, the workflows update the PasswordChangeHistory attribute, which contains the time, the result (success or failure) and the associated user.This attribute should be regularly cleared with the RemoveAccountPasswordChangeHistory consistency rule.

Depending on the supported API technology (Java or C/C), connectors run either in the Java-based Identity Server (IdS-J) (as part of the workflow) or in the C-based Identity Server (IdS-C).The communication between the Java-based workflow and the C++-based

connector is performed via SOAP/HTTP in a synchronous mode.

All workflows use common services for logging, auditing, statistics and retry.Auditing and statistics information is also stored as status entries in the DirX Identity status area.See the section "Understanding Password Synchronization Workflow Operation" for more information.

## 5.2. Password Changes from Windows Domains

Users can change passwords with standard Windows methods at any Windows domain controller.The next sections describe how DirX Identity password management components process password changes from Windows domains.

### 5.2.1. Password Changes via the Windows Password Listener

The following figure illustrates how the DirX Identity password management components interact to process password changes from Windows users and distribute them to the relevant target systems.

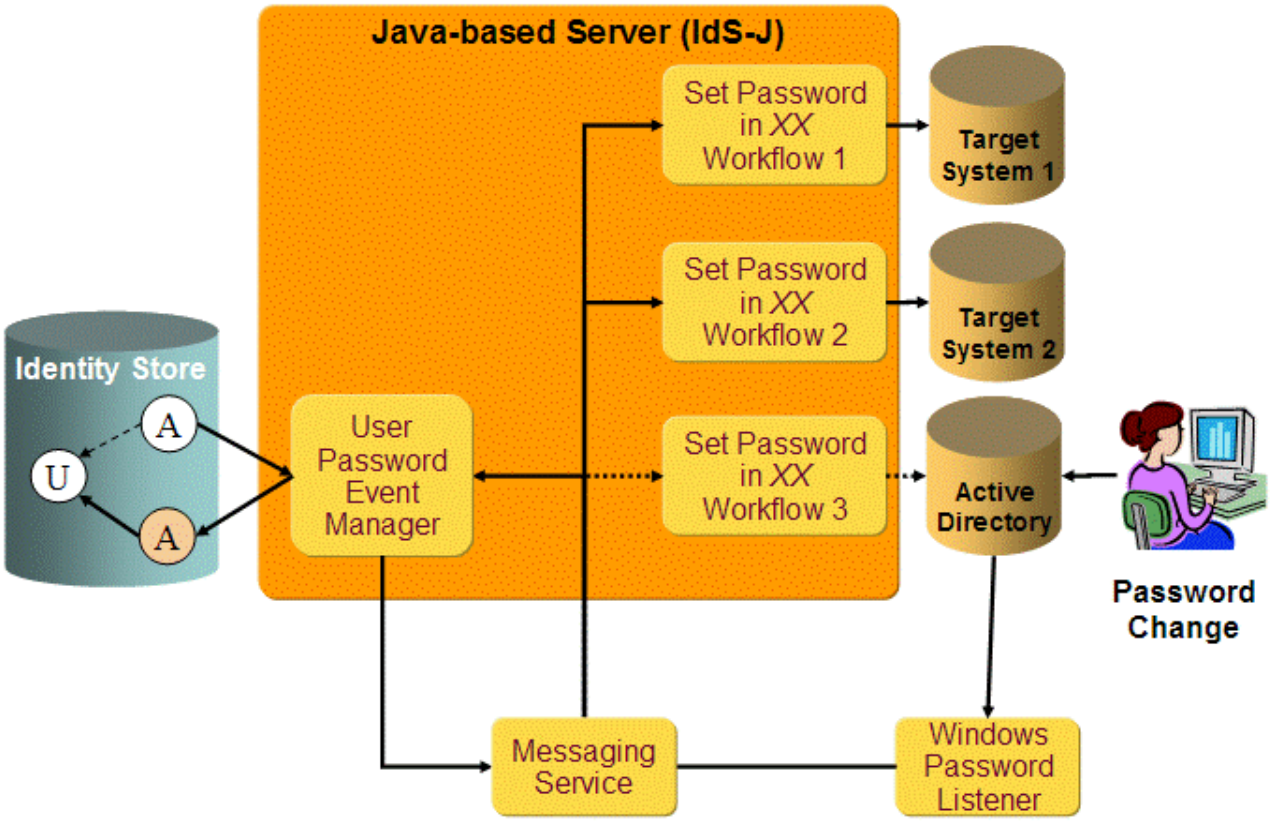



*Figure 16. DirX Identity Windows Password Listener Component Operation*

As shown in the figure, the detailed sequence of process steps is:

- The user changes his password in the Windows environment.
- The password change is sent as a Java Messaging Service (JMS) message to the DirX Identity Messaging Service. It contains the Windows user account name as well as the domain and forest information.

- A User Password Event Manager is started in the Java-based Server (IdS-J) when such events arrive.
- If a password change event arrives, the User Password Event Manager retrieves the relevant fields from the message. In this case, the User Password Event Manager identifies first the corresponding target system in the Identity Store with the help of the domain and forest information (these fields correspond to the fields in the Advanced tab of the target system object). Next, the manager searches the account within this target system.
- If it locates the account, the User Password Event Manager follows the link to the corresponding user entry in the Identity Store and changes the password at the user entry. If it cannot locate the account, the manager stores the event for a configurable number of retries. It sends a notification event if it reaches the retry limit without successfully locating the account.
- Using the information in the user entry, the manager retrieves all accounts that belong to this user that must be updated with the new password. For each account, the manager creates a request to change the password in the corresponding target system. The target system that generated the password change event is omitted (this would result in cyclic behavior otherwise).
- Set Password in *XX* workflows are started immediately to process the requests. If the change is successful, the request is deleted. If it is not successful, a configurable number of retry cycles are issued. If all fail, a notification event is sent. In all cases, the workflows update the PasswordChangeHistory attribute at the accounts.
- Target system interfaces that cannot be accessed via Java interfaces but in C or C technology are handled by connectors running in the DirX Identity C-based Server (this is not shown in the previous figure but is similar to the "Password Changes from Web Applications" section). A Set Password in *XX* workflow runs in the Java-based Server, produces a synchronous SOAP/HTTP request for the password change connector in the C++-based Server and waits for a synchronous response. If successful, the request is deleted. If not, a configurable number of retries is performed. If all fail, a notification event is sent.

For more information, see the section "About the Windows Password Listener" and the description of this component in the *DirX Identity Connectivity Reference*.

### 5.2.2. About the Windows Password Listener

The Windows Password Listener is a DirX Identity agent that captures passwords in clear text from the Windows domain controller and transfers them to the messaging service where the DirX Identity event manager picks them up for further processing.

The Windows Password Listener agent is implemented as a separate tool that does not need DirX Identity server components to be installed on the same machine.It must be installed and registered on each domain controller in the network.

When the Windows domain controller is started, it calls the Windows Password Listener’s initialize method, which sends a message to the event manager to retrieve the public encryption key.On the other hand, if the public encryption key is exchanged, the Java-based Server distributes the changed key to all Windows Password Listeners.

The Windows Password Listener captures password at the Windows domain controller, encrypts it with the public encryption key and transfers it as message to the messaging service (the message serves as an event and transfers the data in parallel).The message is marked with the originator **APETA://ADS/***domain*.It includes the forest name, domain name, computer name, userName, and password (encrypted) attributes and the flag 'User must change password during next login'.

The *DirX Identity Connectivity Reference* provides more details about the Windows Password Listener.

## 5.3. Password Changes from Web Applications

Users can change passwords using Web applications.The standard method is to use the DirX Identity Web Center; for a detailed description see the *DirX Identity Provisioning Administration Guide*.As an alternative to Web Center, you can integrate the Java classes of the DirX Identity Web Event Trigger into your Web applications; for more information, see the corresponding section in the *DirX Identity Connectivity Reference*.

In both cases, the Web application sends a password change message to the DirX Identity Messaging Service for further processing.The following figure illustrates the event and how the DirX Identity password management components interact to process it.

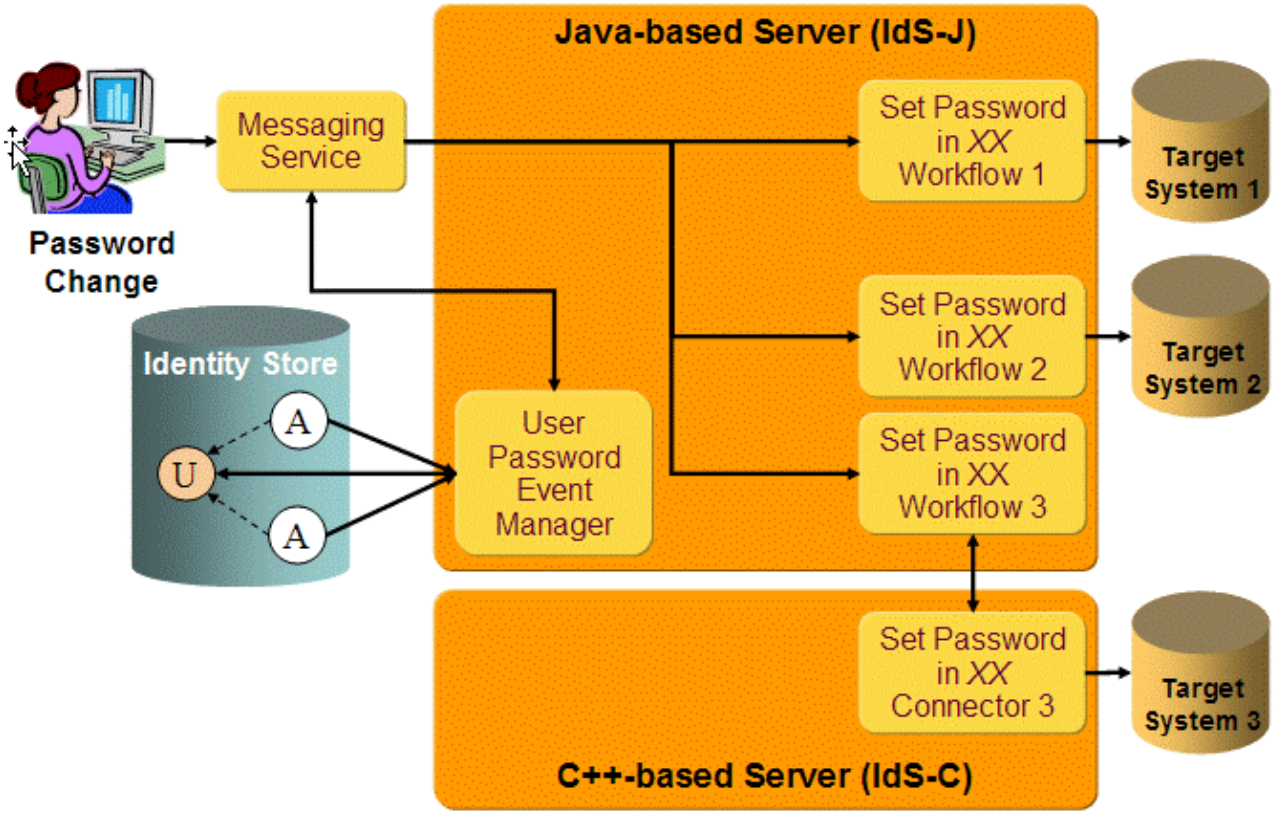



*Figure 17. Web-based Password Change Processing*

As shown in the figure:

- The user changes his password through the Web application.

- A user who is working with the specialized Web Center for Password Management can de-select accounts from password change processing; passwords for these accounts will not be updated.
- The password change is sent as a Java Messaging Service (JMS) message to the DirX Identity Messaging Service.
- A User Password Event Manager workflow is started in the Java-based Server (IdS-J) when such events arrive.
- If a password change event arrives, the User Password Event Manager retrieves the relevant fields from the message.The distinguished name (DN) of the user allows the User Password Event Manager to identify the user entry in the Identity Store.
- If the User Password Event Manager can locate the user entry, it changes the password at the user entry.Otherwise, it stores the event for a configurable number of retries.After several unsuccessful retries, the User Password Event Manager sends a notification event.
- Using the information in the user entry, The User Password Event Manager retrieves all accounts that belong to this user and that need to be updated with the new password, excluding those accounts that the user previously de-selected.For each account, the manager creates a request to change the password in the corresponding target system.
- Set Password in *XX* workflows are started immediately to process the requests.If the change is successful, the request is deleted.If it is not successful, a configurable number of retry cycles are issued.If all fail, a notification event is sent.In all cases, the workflows update the PasswordChangeHistory attribute at the accounts.
- Target system interfaces that cannot be accessed via Java interfaces but in C or C technology are handled by connectors running in the C-based Server.A Set Password in *XX* workflow runs in the Java-based Server, produces a synchronous SOAP/HTTP request for the password change connector in the C++-based Server and waits for a synchronous response.If successful, the request is deleted.If not, a configurable number of retries is performed.If all fail, a notification event is sent.

## 5.4. Configuring the Password Synchronization Workflows

There are two types of password synchronization workflows:

- A **User Password Event Manager** workflow, which handles password change events published by Windows Password Listener or Web Center, updates the user password in the Identity Store and creates password change requests for all the user's accounts.
- A **Set Password in *XX*** workflow for each target system instance, which updates the account password in the target system according to the change request issued by the Password Change Event Manager workflow.

The structure of the password synchronization workflow configurations is very similar.All password synchronization workflows consist of one mandatory activity to update the password and an optional error activity, which notifies the affected user of failed password updates.

Each workflow is responsible for a specific family of events, which must be set in the **Is applicable for** section of the first tab of the workflow configuration entry, named "Workflow".

You must set the source of password change events for the User Password Event Manager. By default, there is only one manager that listens to all password change events. This is indicated by the wildcard "*" in all fields. For load distribution, you can deploy multiple User Password Event Managers, each listening for a different set of events.

In each change password request, the User Password Event Manager workflow sets a message topic, which includes the following fields from the target system entry: **Type**, **Cluster**, **Domain**. They are labeled differently in some target system types; for example, forest and domain for Windows systems. The Java-based Server finds the appropriate Set Password in *XX* workflow by matching the topic with the settings of the Type, Cluster and Domain fields in the workflow configuration.

A Set Password in *XX* workflow must be configured for each target system instance where passwords are to be synchronized. In order to associate the workflow and the target system, the fields Type, Cluster and Domain of the workflow configuration must match the corresponding fields of the target system entry.

The rest of the configuration is almost the same for all workflows.

Set the flag to write audit logs, enter the number of retries in case of temporary failures and the waiting time between retries.

Select the connected directory at which to update the password and a bind profile. The only exceptions are target systems, which are accessed by a C/ C# based connector. They run in C-based Servers. Therefore, the workflow configuration needs the C++-based Server and the connector within the server in order to know where to send the SOAP requests.

The **resource family** indirectly determines on which servers an activity may be run. It tells the system which resources it needs for processing. It is a good idea to associate resource families with target system types or target system instances, if there are a number of them. Choose a resource family "LDAP" to be associated with target systems of type LDAP, a resource family "ADS" to be associated with Windows systems, and so on. Setting the activity resource family "LDAP" instructs the system to run this activity only on servers that provide access to this type of resource.

Java-based Servers provide resources; for example, LDAP.Activities run only on servers that are associated with the same resource that the activity requires.An activity with a resource family LDAP needs a server that has a reference to the resource family LDAP.

**Make sure that, for each activity, there is at least one Java-based Server associated with the same resource family!**

The configuration of the error activities is the same for all workflows.With the flag "enabled" set, you decide that the workflow includes the error activity.Otherwise, the workflow contains only the "set password" activity.In this case, permanently failed requests are placed in the dead letter queue.See the chapter "Using Web Admin" in the *DirX Identity User Interfaces Guide* for information about how to view, re-process and delete these

entries.

With the other options, you set the mail fields: from, to, subject, and body.

Even if you have configured a Set Password in *XX* workflow for your target system, you need to enable the target system for password synchronization.Reset the flag **Disable Password Sync(hronization)** at the target system object in the Provisioning view group to enable password synchronization.

## 5.5. Understanding Password Synchronization Workflow Operation

The following figure illustrates how the password synchronization workflows operate in more detail.

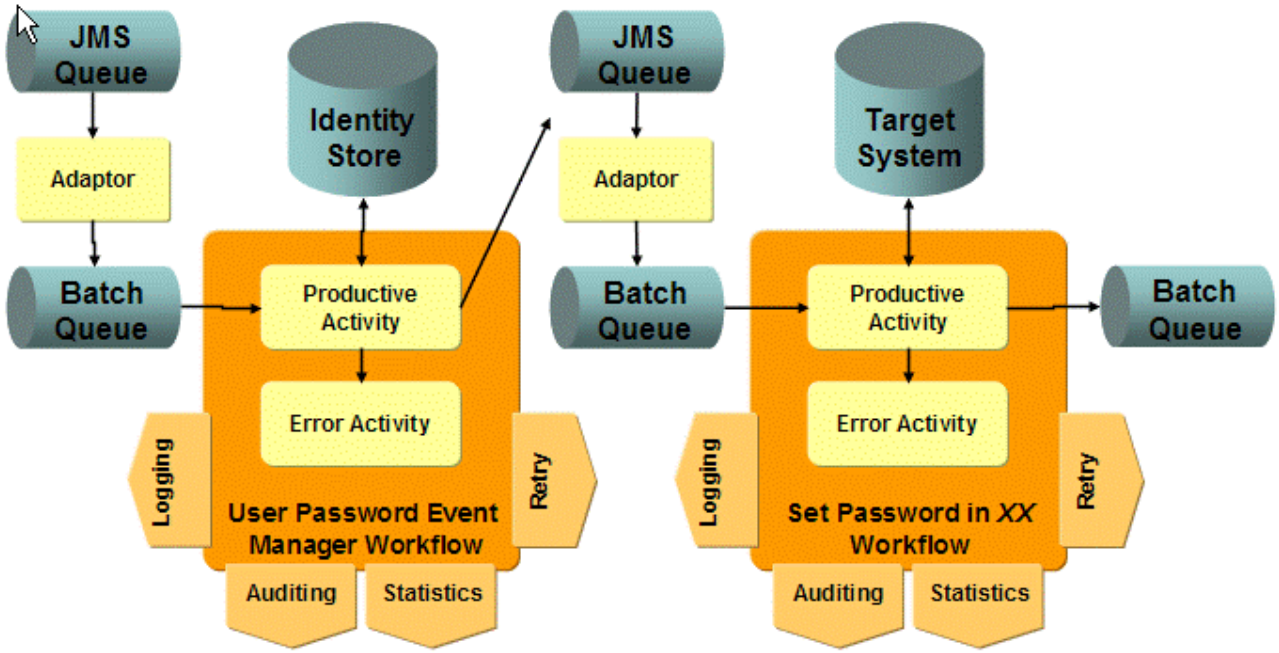



*Figure 18. Password Synchronization Workflow Components*

As shown in the figure, the sequence of steps is as follows:

- The corresponding adaptor (**Password Change Listener**) reads events from an external queue (for example, a JMS queue).It adds the events into its own repository (persistent queue) and then into the memory-based **Batch Queue** and deletes them from the external queue.
- A **Workflow Dispatcher** component (not shown in the figure) analyzes the events in the Batch Queue and starts the workflows that can handle these types of events.In this case, the password event manager workflow (**User Password Event Manager**) is started.'
- After start, the **User Password Event Manager** workflow analyzes the event's content and tries to identify the relevant user in the Identity Store to which this password change event belongs:

- If successful, the User Password Event Manager workflow changes the password of the user, but only if the new value and the old one does not match. If they are identical, the workflow stops processing this event and does not request changes for the accounts. This method effectively prevents password change loops. See the section "About the Password Change Algorithms" for details.
- If the workflow successfully changes the user password, it creates setPassword requests for all accounts of this user where the password shall be changed and sends these to the internal temporary SetAccountPasswordListener JMS queue for further processing by the relevant **SetPassword in *XX*** workflows. These requests include all account attributes that identify the account in the connected system and the changed password.
- If unsuccessful, the User Password Event Manager writes the event to the retry channel. After some waiting time, the Event Manager processes these events again until their retry limit is reached. See the section "Error Handling and Retry" in "Managing Java-based Provisioning Workflows" for details.
- Events that fail even after retries are passed to the **Error Activity**, which sends an e-mail to the affected user. See the section "Error Handling and Retry" in "Managing Java-based Provisioning Workflows" for details.
- In all cases, the User Password Event Manager workflow writes logging, auditing and statistics information to be processed by the corresponding handlers.
- At the end of the workflow, successful or not, the requests are removed from the adaptor's repository.
- The **Set Account Password Listener** adaptor reads the events from the **JMS Queue** and packs them into the **Batch Queue**.
- The **Workflow Dispatcher** analyzes the (set password) event and starts the **Set Password in *XX*** workflow that is configured for this target system instance.It associates the event and the workflow by their type, domain and cluster attributes.
- After start, the **Set Password in *XX*** workflow reads the event from the JMS Queue:
- Its first activity constructs the account identification in the target system and issues the password change request to the target system.
- The workflow engine retries unsuccessful requests until the retry limit is reached, and then passes the requests and their responses to the error activity, which issues e-mail notifications to the affected user.See the section "About the Password Change Algorithms" for details.
- In all cases, the Set Password in *XX* workflow writes logging, auditing and statistics information to be processed by the corresponding handlers.
- At the end of the Set Password workflow, successful or not, the requests are removed from the JMS Queue.

## 5.6. About the Password Change Algorithms

This section provides details about the password change algorithms used by the DirX Identity password management components.

To minimize unnecessary password changes that could result in refused updates at the target system side due to password history mechanisms, DirX Identity checks each password change to determine whether or not the new password is identical to the previous one.It uses the password history attribute for this task even if the password history policy is not enabled for a specific user.

The algorithm for a user with enabled password history is shown in the following flowchart.

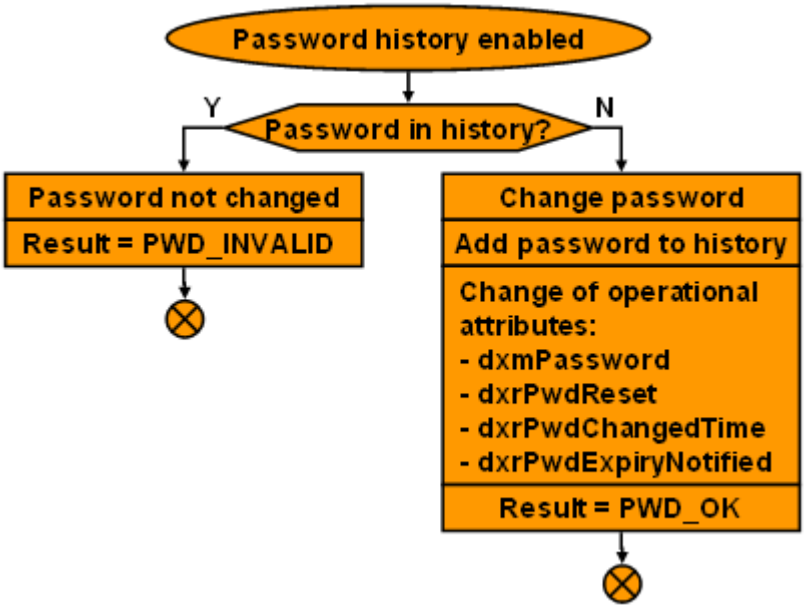



*Figure 19. Password History Algorithm (Password History Enabled)*

In this algorithm, DirX Identity changes the password if it is not already contained in the password history. The history is updated accordingly. Otherwise the password change is rejected.

The algorithm for a user with disabled password history is shown in the following flowchart.

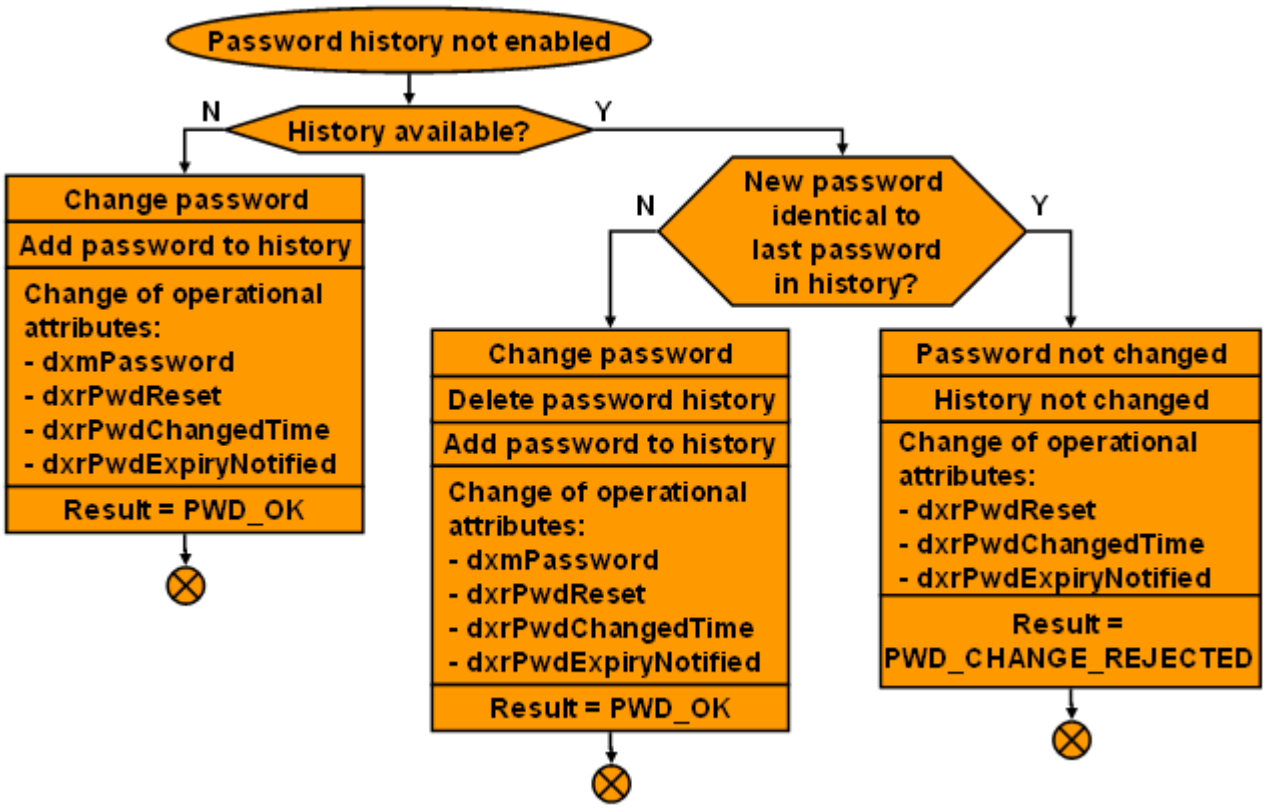



*Figure 20. Password History Algorithm (Password History Disabled)*

In this case, DirX Identity changes the password if no password history is available. The password value is stored in the password history for further checks.

The test to determine whether the new password is identical to the existing one is made against the stored value in the password history. If the password is identical, the change is rejected. If it is not identical, the password is changed and the password history value is replaced with the new one.

The test relies on the fact that the new password is stored in the password history even if the password history policy is disabled for the user. The password policy, however, can also be configured to prevent storage of the new password if the history is disabled. In this case, the check for identical passwords will always fail, causing the new password to be processed as if it was different from the previous one. Since this will cause some issues in a system with one or more Windows Password Listeners, storing the new password must not be disabled in such a system.

# 6. Managing DirX Identity Servers

This chapter explains how to manage:

- The Message Broker
- The Java-based Identity Server (IdS-J)
- The C++-based Identity Server (IdS-C)

It also provides information on:

- Distributed deployments and scalability
- High availability and recovery
- Diagnostics
- How to manage daylight savings time
- Connector frameworks

For information on high availability for both Java- and Tcl-based workflows, please see the use case document *High Availability*.

## 6.1. Managing the Message Broker

DirX Identity uses a JMS Message broker for most of its internal communication:

- The DirX Identity services (for example, the privilege resolution service), Web Center, Java-based real-time workflows and metacp to send real-time events for immediate processing by Java-based workflows.
- The Windows Password Listener and Web applications, to send password change events to be processed by password event managers and password synchronization workflows.
- The C++-based Identity Servers (IdS-C servers) in a distributed environment, to exchange the command messages that start and control Tcl-based workflows. The same interface is used by the **runwf** tool.
- The C++-based DirX Identity Status Tracker component, to receive messages from various components and create status entries in the status area.

The **Message Broker** is one or more Apache ActiveMQ instances.The ActiveMQ instances must be installed via the DirX Identity installer to manage locations and names, but the target servers for these instances are not predefined, so that the messaging system is flexible and can scale easily.

This section describes how to plan for and set up the message broker, including:

- Deployment options (single broker, high availability)
- Installation, configuration, start/stop, JMX access and logging
- Basic concepts of how DirX Identity uses the message broker

- Messages and message sizes used in DirX Identity

### 6.1.1. Planning the Message Broker Deployment

You can deploy the Message Broker in different ways, depending on your load-balancing and high-availability requirements. Installation/configuration wizard deployment options include:

- One Message Broker instance installed on the same system as an IdS-J or IdS-C server
- One Message Broker instance installed on an external server
- Multiple Message Broker instances spread over IdS-J / IdS-C and external servers sharing the same database for persistent messages (database on a shared drive)

Apache ActiveMQ offers various options on how to operate multiple instances. The DirX Identity configuration uses the following implementation:

- Only one Message Broker instance is accessible for clients. This broker has exclusive access to the database (DB lock) for persistent messages.
- All other instances are up and running, but can't access the database. If the exclusive broker is unavailable, the next instance takes over the database access, captures the persistent messages and starts up the connectors to be accessible for the clients.
- The switch from one broker instance to another instance is transparent for the clients. DirX Identity's Message Broker name and instance management service handles the discovery of the Message Broker instance.
- To ensure the fail-over capability, the database for persistent messages must be on a shared drive to which all broker instances have access. For a single-broker installation, you can install the database locally.

### 6.1.2. About the Message Broker Components

Message broker components include:

- The Message Broker wrapper container
- The database for persistent messages
- The start/stop service/daemon

Each Message Broker runs its own wrapper container which is deployable on Windows and/or UNIX systems.

Each Message Broker needs a database for persistent messages. The built-in database is "KahaDB". The database is installed by the DirX Identity installation. The location of the database depends on the message broker deployment in use.

Each Message Broker is represented by a service on Windows or a daemon process on UNIX. Service names are:

**DirX Identity Message Broker** *number* (Windows)

**ids-mbrk-***number* (UNIX)

The number is assigned during Message Broker configuration. Only one Message Broker per server is supported. In a high availability scenario, you may have multiple Message Brokers across a distributed DirX Identity installation.

### 6.1.3. Starting the Message Broker

The services that make up the Message Broker normally start automatically when the system is booted. The service starts independently of the IdS-J / IdS-C services. Message broker services include:

- Service name on Windows: **DirX Identity Message Broker** *number*
- Process name on Windows: **wrapper.exe**
- Process name on UNIX: **wrapper**
- Controlled processes on Windows: **java.exe**
- Controlled processes on UNIX: **java**

The shell script *install_path***/etc/dmmbrk-***number* starts the service on Linux when the system runs in multi-user mode.

### 6.1.4. Configuring the Message Broker

The DirX Identity-supported configuration options for the Message Broker are:

- Target server for a Message Broker instance
- Location of the database for persistent messages
- Optional transport options
- Optional fail over options in case of a High Availability deployment. They are to be configured at the parent entry of the message broker configuration entries. For details see the Active MQ documentation: http://activemq.apache.org/failover-transport-reference.html.

### 6.1.5. Monitoring the Message Broker

Use the DirX Identity Server Admin to monitor a Message Broker. Server Admin provides an overview of installed DirX Identity servers and components including the Message Broker instance(s). The instance status and a link to the Web console are provided in this view.

ActiveMQ provides a Web Console, which allows you to view the number of messages in the queues and even the messages themselves. The port can be configured as admin port in the associated system service of the broker entry. For more details see the ActiveMQ web page (http://activemq.apache.org/web-console.html).

To call the Web Console, use the following URL:

**http://localhost:8161/admin**

or in case of SSL configuration:

**https//*your installation host***:8161/admin*

Port number **8161** is the default. In the ActiveMQ Message Broker configuration step, you can set a different port number.

For more information, see the ActiveMQ web page (http://activemq.apache.org/web-console.html).

### 6.1.6. Message Broker Logging

When the Message Broker starts for the first time, default log levels are applied. They ensure that error and warning logs of all components are written. The log files are stored in the data folder of the local broker's home directory (*install_path*/**messagebroker**). Log file names start with **wrapper** (for the service handler) or with **activemq** (for the broker itself).

### 6.1.7. Message Broker Instance Naming

Using multiple Message Brokers requires an easily-understood naming scheme for service names, LDAP entry configuration names and file folder names.

Message Broker Object Naming

The Message Broker objects in the Connectivity view group (Connectivity → Messaging Services) follow this naming scheme:

**Message Broker** *number*

where *number* is the number assigned to the Message Broker. The numbers are assigned dynamically; the first Message Broker to be configured is assigned the number 1.

Example for a server object:

Message Broker 1

Service Naming

The services on Windows use the following naming scheme:

**DirX Identity Message Broker** *number*

Example for a server:

DirX Identity Message Broker 1

**File Folder Naming**

The file folder in the installation area is *install_path*/messagebroker* because only one Message Broker per local installation is supported.

### 6.1.8. JMX Access to the Message Broker

By default, JMX access to the Message Broker needs password authentication. Authentication is performed by passing the user credentials to an LdapLoginModule which tries to bind to the Connectivity LDAP server with these credentials. When the bind is successful, the JMX access is successfully authenticated.

Per configuration, the LDAP user is **cn=DomainAdmin,cn=domain,dxmC=Users,dxmC=DirXmetahub**.

On the JMX client-side, you only need to give the “DomainAdmin” as the username and the appropriate password. This is configured in the file **jmxldap.cfg** in the Message Broker’s **conf** folder.

If the SSL flag is activated in the system-wide configuration, the JMX access is also secured by SSL. In this case, a non-SSL access is not possible.

If SSL is configured for the Connectivity store, the authentication process to the LDAP server is also performed using an SSL connection. In this case, you need to make sure that the LDAP server’s root CA chain certificates are in the **cacerts** file of the Identity Java environment.

### 6.1.9. Understanding the Java Messaging Service

DirX Identity uses both message paradigms supported by Java Messaging Service (JMS): point-to-point (P2P) and publish / subscribe (Pub/Sub).

In P2P messaging, a producer sends a message to a queue. Only one or several consumers can read them from this queue. The broker passes a message to only one of them. In Pub/Sub a producer sends a message with a topic. One, several or even no consumers can subscribe to that topic (then they are called "subscriber"). Each of them receives its own copy of the message. When no consumer has subscribed to the topic, the Message Broker simply deletes it.

Both P2P and Pub/Sub allow message producers and consumers to remain independent of one another. No producer needs to know the consumer(s) of its messages, and a consumer does not need to know the producer(s). You can have multiple producers and multiple consumers. Consumers process the message when they have time for it. The producer must not wait until the consumer is finished; that is, messages are processed asynchronously.

Messages can be persistent or transient. When a message is declared persistent, the consumer(s) need not be online when the producer sends it. For Pub/Sub, as soon as the consumer has subscribed to a topic, it receives all messages sent after that moment even if it stops and starts again later. For P2P, persistent messages are stored in the queue until some consumer reads and acknowledges them. Transient messages are lost when the Message Broker stops.

## 6.1.10. Using Messages in DirX Identity

Most of the DirX Identity messages are persistent, so they are not lost when a Message Broker or a consumer is down. Only very few message types are marked as transient: they are mainly to distribute configuration update notifications or certification information. As the servers read their configuration on start-up and the Windows Password listener requires configuration and certification updates also on each start-up, no information is lost.

Some of the DirX Identity components work in P2P, others work in Pub/Sub mode.

All the messages around real-time workflows, especially password changes, are P2P. Only one consumer is intended to process them.

Pub/Sub messages are used for:

- Controlling and monitoring Tcl-based workflows by the C-based Identity Server. They are produced by the components of the C-based Server itself (scheduler, workflow engine, agent controller) and the Identity Manager. The meta controller **metacp** can also produce Pub/Sub messages, but they are no longer used for the standard workflows.
- Issuing update notifications, mainly of configuration and of certificates. The main consumer is the Windows Password Listener in order to get up-to-date information on available Message Brokers and on current certificates.

#### 6.1.10.1. Queues of the Java-based Server

The message queues described here are consumed by the Java-based Server adaptors to process real-time events. Queue names are case-sensitive and handled in lowercase internally.

Note that the domain prefix is optional. It depends on the flag **Include domain into topic** of the **General** tab of the domain object.

*domain***.dxm.event.pwd.changed**:

A message in this queue indicates the password change for a user.

*domain***.dxm.event.svctsaccount.pwd**:

A message in this queue indicates a password change for an account or that the password of an account has expired.

*domain***.dxm.setpasswordrequest**:

A message in this queue starts a password provisioning workflow to set the password of an account in a connected system.

*domain***.dxm.setpasswordrequest._default**:

This queue contains those messages sent to *domain***.dxm.setpasswordrequest** that can be

processed in any Java-based Server.

*domain***.dxm.request.provisiontots**:

A message in this queue triggers the synchronization of account, group and membership changes from target systems to connected systems.

*domain***.dxm.request.provisiontots._default**:

This queue contains those messages sent to *domain***.dxm.request.provisiontots** that can be processed in any Java-based Server.

*domain***.dxm.event.ebr**:

A message in this queue indicates the change of a user, business object, account or other domain entry and is processed by an Event-based Maintenance workflow.

*domain***.dxm.request.workflow.ebr**:

A message in this queue starts a maintenance workflow by its DN; for example, the certification campaign controller.

*domain***.dxm.request.importtoidentity**:

A message in this queue indicates a change in an entry in a remote system (for example, a user in an Active Directory) that needs to be imported into the DirX Identity domain.

*domain***.dxm.request.importtoidentity._default**:

This queue contains those messages sent to **domain.dxm.request.importtoidentity** that can be processed in any Java-based Server.

*domain***.dxm.request.workflow.provisioning**:

A message in this queue is used to start a real-time workflow by name (used by Identity Manager and Scheduler).

*domain***.dxm.request.workflow.provisioning._default**:

This queue contains those messages sent to *domain***.dxm.request.workflow.provisioning** that can be processed in any Java-based Server.

*domain***.dxm.request.workflowengine**:

A message in this queue is used to request the request workflow engine to update the workflow.

*domain***.dxm.request.activitytask**:

A message in this queue is used to start an activity in a request workflow.

*domain***.dxm.notify.mail**:

A message in this queue is used to send an email.

*domain***.dxm.notify.sms**:

A message in this queue is used to send an SMS.

*domain***.dxm.request.user.resolve**:

A message in this queue identifies a user. It will be resolved by the Resolution Adapter.

The following queues are created on demand when the Provisioning workflows for a connected system should run only on dedicated servers:

*domain***.dxm.request.provisiontots.***target system identifier*:

This queue contains those messages sent to *domain***.dxm.request.provisiontots** that refer to entries of the selected target system.

*domain***.dxm.request.workflow.provisioning.**target system identifier:

This queue contains those messages sent to *domain***.dxm.request.workflow.provisioning** that refer to entries of the selected target system.

*domain***.dxm.setpasswordrequest.***target system identifier*:

This queue contains those messages sent to *domain***.dxm.*setpasswordrequest** that refer to entries of the selected target system.

*domain***.dxm.request.importtoidentity.target system identifier**:

This queue contains those messages sent to *domain***.dxm.request.importtoidentity** that refer to entries of the selected connected system.

For information on the target system identifier and when these queues are created, see the section "Distributing Deployments and Scalability" → "Separating Traffic for Selected Connected Systems".

#### 6.1.10.2. Topics of the Java-based Server

Messages to the following topics are consumed by one or more Java-based Servers. Each Java-based Server automatically subscribes to the following one:

**dxm.event.configuration.changed**:

The DirX Identity Manager publishes a message to this topic when you request to "Load IdS-J Configuration". It triggers a reload of all workflow and schedule definitions. If the domain is not specified in the message body, all servers load their workflow and schedule definitions.

For the following topic, exactly one Java-based Server per domain is intended to subscribe. Select the server by moving the corresponding adaptor to the desired server. You can do this with Identity Manager and right-click on "Manage IdS-J Configuration" or use Server

Admin.

*domain***.dxm.request.configuration**:

Windows Password Listener (WPL) publishes a message with such a topic in order to obtain information about all available messages services and to obtain a new certificate. The list of available message services allows the WPL to switch to another messaging service in case of failure.

If still older Windows Password Listeners (version older than V8.3) are running in the customer environment, the following is relevant. For the following topic, exactly one Java-based Server across all domains is intended to subscribe:

**dxm.event.certificate**:

Events of this type allow the older Windows Password Listeners to obtain new certificates. On start-up, the WPL publishes a message with the topic **dxm.event.certificate.request** to request the current certificate. The Java-based Server returns the requested certificate in a message with the topic **dxm.event.certificate.changed**. WPL subscribes to that topic and thus receives the certificate. If the certificate is changed, the Java-based Server sends the same message to the topic **dxm.event.certificate.changed** so that all Password Listeners are informed immediately.

#### 6.1.10.3. Topics of the C++-based Server

The following message topics are used by the C++-based Server for internal and external communication:

**dxm.command.***machine_name*

This event type represents commands for the workflow engine (create or start workflow instance; create or start activity instance).

**dxm.fileservice.***machine_name*

Events of this type allow transferring files via the JMS messaging service.

**dxm.statustracker**

Various components of the C++-based Server create status information with this event type. The status tracker processes this information and creates the according status entries in the Monitor View.

#### 6.1.10.4. Topics of the Password Listener

The following message topic is used by the Windows Password Listener communication:

*domain***.dxm.response.configuration**:

Windows Password Listener (WPL) receives messages with such a topic in order to obtain information about all available messages services and to obtain a new certificate.

#### 6.1.10.5. About Message Sizes

The size of the sent messages varies for the different kind of messages.

##### 6.1.10.5.1. Password Messages

The password messages are about 1250 bytes each. The reason is that the encrypted password is converted to base64 format.

##### 6.1.10.5.2. Real-time Events

The real-time event messages are about 700 bytes each. Here is an example (without the message frame):

```
<spml:modifyRequest xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
xmlns:spml="urn:oasis:names:tc:SPML:1:0"
xmlns:event="urn:siemens:dxm:EVENT:1:0"
xmlns:order="urn:siemens:dxm:ORDER:1:0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
requestID="svc.modify.dxm.request.provisionToTS.LDAP.cluster=
'localhost'.
resource='cn=My-Company'.uid-8b19a143--29c9507c-1132f75a2b8--7fff">
<spml:identifier type="urn:oasis:names:tc:SPML:1:0#DN">
<spml:id>cn=Alexander Gerber 5217,cn=Accounts,cn=Extranet
  Portal,cn=TargetSystems,cn=My-Company</spml:id>
</spml:identifier>
<spml:modifications/>
</spml:modifyRequest>
```

The messages contain only the identifier of the object. The workflow reads the rest of the information from the LDAP directory.

##### 6.1.10.5.3. Command and Status Messages

For a Tcl-based workflow with two activities started from the DirX Identity Manager, the size of the messages in bytes is approximately:

| Message Type | Number of Bytes |
|---|---|
| create | 640 |
| create acknowledge | 540 |
| execute | 650 |
| execute acknowledge | 540 |
| destroy | 370 |
| **Sum** | **2740** |

| Message Type | Number of Bytes |
|---|---|
| + 8 status messages<br>(800 through 1050 bytes per status message) | 8400 |
| **Total** | **11140** |

If the workflow is started from the scheduler in a non-distributes environment, only status messages are sent via the messaging service. All other messages (command messages) are sent via an internal queue mechanism. In this case, the total number of bytes for a two-step batch workflow is about 7400 bytes.

When using compression mode, the total number of bytes (and thus the system load) can be significantly reduced:

| Compression Mode | Number of Messages | via Manager | via Scheduler |
|---|---|---|---|
| None | 8 | 10150 | 7400 |
| Compressed | 5 | 7375 | 4625 |
| Minimized if OK | 1 | 3675 | 925 |
| Suppressed if OK | 0 | 2750 | 0 |



"via Scheduler" means that the batch workflow runs in a non-distributed environment.

#### 6.1.10.5.4. File Transfer Messages

The overhead for a file transfer message is 660 bytes plus the filename length (if the file name is 20 bytes, the overhead is 680 bytes).

There is a message length parameter in the messaging service object which is 1 MB by default. You can set this value from 32 K to 4 MB (other values outside of these boundaries are set automatically to these boundaries).

If a file is 1.5 MB and the file name is 20 bytes, then the message is divided into a block with 1 MB - 680 bytes and another one with 1.5 MB - (1 MB - 680 bytes). Plus overhead the first message is exactly 1 MB, the second one is 0.5 MB + 1360 bytes.

So the general formula is:

limit = (configured value)
overhead = 660 + nameLenght
blocksize = limit - overhead

The last block can be smaller.

### 6.1.10.6. Messaging Subscriptions

JMS Topic subscribers must use a unique name. This enables the Message Broker to distinguish different subscribers when they re-connect. DirX Identity components use the following naming scheme.

#### 6.1.10.6.1. Java-based Server

*adaptorname*

where

**adaptorname**
is the name of the adaptor that uses this subscription.

Examples:

```
AdminRequestHandler
ConfigurationHandler
CertificateHandler
```

#### 6.1.10.6.2. C++-based Server

*hostname*.**jms.dxmmsssvr_dxm.***component.servername*

where

**hostname**
is the hostname where the server is running.

**component**
is the component that uses this subscription:

**command**
used to start Tcl-based workflows.

**statustracker**
the status tracker receives status messages that are written to the LDAP directory.

**fileservice**
used by the file service to exchange data files between machines.

**servername**
the name that is defined in the **dxmmsssvr.ini** file (attribute dnServerName).

For the Status Tracker, it is just **jms.dxmmsssvr_dxm.***component*

Example:

```
myhost.jms.dxmmsssvr_dxm.command.myhost
```

#### 6.1.10.6.3. Meta Controller

The meta controller allows sending JMS messages with this naming scheme:

*hostname*.**jms.metacp.***hostname****dxm._*****component.identifier**

where

**identifier**
a unique identifier.

For a description of the other fields, see the description of the C++-based Server.

Example:

```
myhost.jms.metacp.myhost_dxm.command.c0a860862gui
```

#### 6.1.10.6.4. Manager

The Manager creates for the **Process Table** feature only non-durable subscriptions that use this naming scheme:

**gui.**_identifier1.username_identifier2

where

**identifier1**
is a unique identifier of the manager instance.

**username**
is the username with which the manager was started.

**identifier2**
is unique identifier for the subscriber.

Example:

```
gui.c0a8e680.metatest_a81b2bc71aa9cac5_-1a203253_12ec46696c8_-7fe0
```

## 6.2. Managing the Java-based Server

This section describes the Java-based Identity Server (IdS-J), including information about:

- Java-based server components
- Server processes and how to start and configure them
- Recovery
- Auditing
- Statistics
- Logging
- Naming schemes
- Resource families
- JMX access

### 6.2.1. Server Components

The Java-based Identity Server (IdS-J) comprises a complete infrastructure to run event-driven and scheduled synchronization and request workflows. The following figure shows the Java-based Server components.

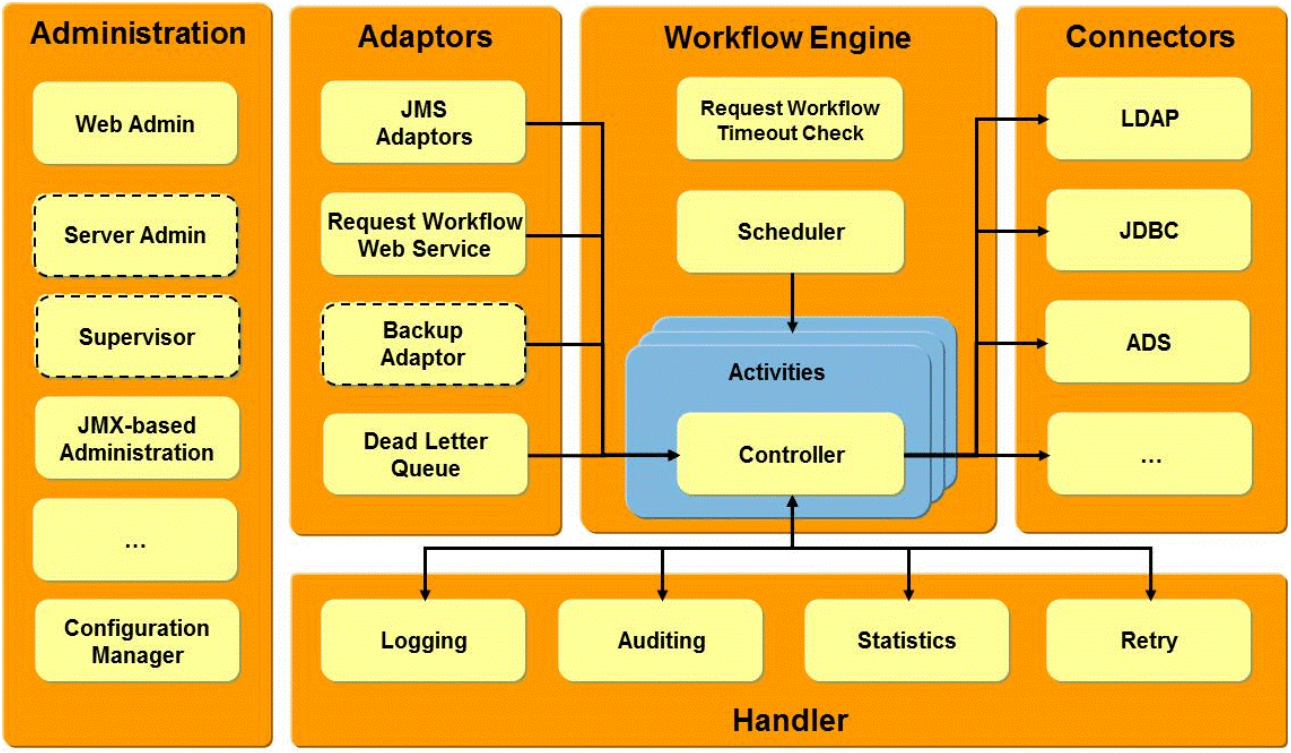



*Figure 21. DirX Identity Java-based Identity Server Components*

The next sections provide an overview of the Administration, Adaptors, Workflow Engine, Connector and Handler components.

#### 6.2.1.1. Administration Components

You can use Web applications or JMX clients via JMX Beans to administer the Java-based Server:

- **Web Admin** - a specialized administrative Web application for monitoring and controlling the Java-based Server in which it is embedded.
- **Server Admin** - a specialized administrative Web application for monitoring and controlling several Java- and C++-based Servers. This application is also running in the embedded Tomcat of each Java-based Server and is only enabled if High Availability was activated.
- **Supervisor** - a servlet running in the embedded Tomcat Web Container that can be started if high availability is activated. It is responsible for monitoring another IdS-J and optionally all C-based Servers. If one of these servers crashes, the supervisor moves its functionality - JMS adaptors, Request Workflow Support, Java Scheduler and Status Tracker - to its local server or another C-based Server (status tracker). It also requests the backup adaptor to recover the backed-up messages.
- **JMX-based administration** - any custom JMX client can be used as an alternative to the

Web applications for administering or controlling the server; for example, Oracle's JConsole.

The **Configuration Manager** is an internal component of each Java-based Server that is responsible for loading all necessary configuration information during server startup. It can also perform an update after an explicit request from administrative interfaces.

#### 6.2.1.2. Adapters

**JMS adaptors** in a Java-based Server read specific events from JMS message queues or subscribe to message topics. A JMS adaptor in a Java-based Server is available for each queue and for each topic. In general, adaptors consuming messages from JMS queues can be started in each Java-based Server and thus support load balancing and scalability. Some of the JMS adaptors subscribing to message topics are automatically activated in all Java-based Servers; some others should only be activated in one server. For more details, see "Managing the Messaging Broker".

**Target system-specific adaptors** process Provisioning workflows that are sent to queues specific for a target system or connected system (for example, for importing entries to the Identity Store) so that slow target systems or those with lot of traffic do not slow down the provisioning of other systems. Dispatchers for the corresponding queues distribute the messages either to the default queue or the appropriate target system specific queue. For more details, see "Queues of the Java-based Server" in "Managing the Messaging Broker".

For supporting high availability, a **backup adaptor** can be started. It is responsible for backing up all messages of all JMS adaptors (except target system-specific adaptors) of its monitored server. In case of fail-over, it can be instructed to send the stored messages to its Message Broker.

The **Dead Letter adaptor** receives messages that couldn't be processed successfully in the local server and stores them to a local embedded database. **Web Admin** allows an administrator to either re-process all or a subset of these messages or to delete them.

For supporting request (approval) workflows, the **Request Workflow Web Services** must be activated in one Java-based Server. They are hosted in the embedded Tomcat Web Container.

The **Resolution Adapter** is responsible for resolving a user's privileges to access rights in connected systems. Whenever a client application adds or removes a privilege assignment or changes an attribute that might affect the user's access rights, it sends a message, and the Resolution Adapter calculates the groups and accounts of that user. The Resolution Adapter is started on every Java-based Server. The number of listeners per server for the "resolution queue" is configured in the central configuration entry of the domain; the default is 2.

#### 6.2.1.3. Request Workflows

The **Request Workflow Web Service** is deployed with every Java-based Server and supports request (approval) workflows. It allows for creating a new request workflow, updating its state, especially performing approval, and suspending and resuming a workflow. It is hosted in the embedded Tomcat Web Container.

A special job called **Request Workflow Timeout Check** (previously named **Full Check**) regularly checks timeouts of request workflows and their activities. If it detects a timeout, it sends a request so that the workflow engine updates the workflow state and, for example, terminates the activity or workflow. This timeout check must run on exactly one server per domain. It is configured in the Connectivity View group of Identity Manager by navigating to a Java-based Server and then selecting **Manage IdS-J Configuration** from the context menu.

#### 6.2.1.4. Workflow Engine and Connectors

The workflow engine controls provisioning real-time, maintenance as well as request workflows for approval. Workflows consist of activities; the workflow engine starts these activities and controls their maximum lifetime. In case of timeout, the workflow engine is responsible for retrying activities after temporary errors, for escalation handling and for properly setting the operational status of both activities and workflows.

Workflow activities of provisioning and maintenance workflows are realized by components built on the connector framework. In addition to the connectors that implement the interfaces to external systems, the important components of the framework are:

- The scheduler, which allows you to schedule workflows for a domain defined by schedules. In this case, the Java-based workflows are not run as real-time workflows triggered by events. They retrieve the necessary data from search definitions. Note that exactly one scheduler per Identity domain must be active. Configure it in the Connectivity View group of Identity Manager by navigating to a Java-based Server and then selecting **Manage IdS-J Configuration** from the context menu.
- The controller (called the **join engine** in Provisioning workflows), which is the central component that controls the behavior of a job: it reads configuration, initiates the components and calls the connectors.
- The DirX Identity connectors, which handle search and update operations with the external connected systems or internal event channels to other activities, audit or other adaptors. These ready-to-use connectors map internal SPML requests and responses to connected system API calls.

Activities of request workflows can be realized based on the connector framework, but are most often independent of it and can even be completely proprietary. They can either be automatic or people activities. The workflow engine is responsible for setting the appropriate states for a people activity and start notification jobs when e-mail is to be sent on start or end of an activity.

#### 6.2.1.5. Handlers

Java-based Server handlers provide common functionality to all Java-based Server components. Handlers are available for:

- Logging, to capture log entries and write them to configurable log files. Several log handlers can be set up with individual log levels and output destinations.
- Auditing, to receive audit entries via the audit channel and write them to a destination.

The default audit handler writes to files. A JMS audit handler sends audit messages to the DirX Audit Message Broker. See "Auditing" below on how to configure them and install the JMS audit handler when necessary.

- Statistics, to store the workflow statistics in the Connectivity Configuration (database).

## 6.2.2. Server Processes

With the DirX Identity Configuration (Wizard), you can set up one or more Java-based Identity Servers (IdS-J) per system. Each server runs as a system process starting threads as needed; for example, to process event-based workflows.

You can run one or more Java-based Servers on the same host for the same Identity domain or for different ones.

### 6.2.2.1. Starting the Processes

On Windows, the Java-based Server processes normally start automatically when you boot your system. They start independently of each other.

The process name for IdS-J on Windows is **ids-j.exe**.

On Linux, the servers start automatically if you have followed the instructions in the *DirX Identity Installation Guide*. In both cases, the process name is **java**.

On start-up, the Java-based Server attempts several times to connect to the directory server that holds the Connectivity configuration. If unsuccessful, it does not proceed; you must re-start the IdS-J Server after the directory server is accessible.

### 6.2.2.2. Configuring the Processes

The Java-based Server is controlled by the following initialization (***.ini**) and password files:

*install_path*/**ids-j-***domain*-**S***n*/**bin/idmsvc.ini** (initialization file for Windows)
*install_path*/**ids-j-***domain*-**S***n*/**bin/runServer.sh** (initialization file and start script on Linux)
*install_path*/**ids-j-***domain*-**S***n*/**private/password.properties** (password file)

#### 6.2.2.2.1. Java-based Server INI File Parameters

The Java-based Server (IdS-J) initialization file *install_path*/**ids-j-***domain*-**S***n*/**bin/idmsvc.ini** contains two sections: **Section [Settings]** and **Section [vmargs]**. This topic describes the parameters contained in these sections.

**Section [Settings]**

This section specifies the following general parameters for the Java-based Server:

- **service** - the name of the service.
- **displayname** - the display name of the service.
- **description** - the description of the service.
- **vm** - path to the Java virtual machine.

- **mainclass** - the main class to start with.
- **workingdir** - the working directory (default: **.**)
- **autostart** - whether or not the service starts automatically (default: **TRUE**):
  **FALSE** - manual startup
  **TRUE** - automatic startup
- **timeout** - the timeout value in seconds after which the Service Control Manager assumes a serious error and stops the corresponding service (default **240**; relevant for Windows only).
- **dxi.java.home.bin** - the path to the **bin** folder of the JRE (Windows only).

**Section [vmargs]**

This section specifies the Java virtual machine arguments for the Java-based Server. Only the parameters of interest to the administrator are noted here; changes here can harm the Java-based Server. (Note that not all arguments are described here, which means that the numbers given on the left side of the list below may vary in your installation):

- **0=-Xmx2G** - the amount of memory used.
  Change this parameter if you need more heap space.
- **4=-XX:+HeapDumpOnOutOfMemoryError** - dumps a heap file on **OutOfMemory**.
  Delete this line if you do not want heap files to be written.
- **16=-Dcom.sun.management.jmxremote.port=40005**
  **17=-Dcom.sun.management.jmxremote.rmi.port=40006**- JMX ports. Note that JMX uses two ports. In the configuration wizard, you define only the first port number. The second is then the first number plus one.
  For JMX access, several other defines are set. Of interest is probably just the definition of supported TLS protocols (see the line with **Dcom.sun.management.jmxremote.ssl.enabled.protocols**)
- **25=-Djava.security.auth.login.config=*_path_to_JavaServer_**/bin/jmxldap.cfg* - LDAP authentication is enabled by default for JMX access.
- **40=-DIDM_LOGFOLDER=*_path-to-log-folder_ - the path to the folder for all the log files of the IdS-J (the default is *../logs**). Note that the folder must exist.

If SSL is globally configured, then keystore and keystore password parameters are activated.

The INI file is used on Windows only.

##### 6.2.2.2.2. Java-based Server Startup Script on Linux

On Linux, the Java-based Server is started via the **runServer.sh** script. This file is also the configuration file for the process. It contains the same parameters as the **[vmargs]** section (see the section "Java-based Server INI File Parameters" for details).

##### 6.2.2.2.3. Java-based Server Password File Parameters

The Java-based Server (IdS-J) reads its passwords from the files:

*install_path*/**ssl/password.properties**

*install_path*/**ids-j-***domain*-**S***n*/**private/password.properties**

These files contain all passwords and PINs necessary for correct operation. The first file contains the password and PINs necessary for the entire DirX Identity installation, while the second file contains the domain-specific passwords and PINs.

During startup, all DirX Identity servers require reading the relevant configuration information from the Identity Store. For authentication, passwords and PINs must be present in the server configuration files. The servers can read passwords or PINs in clear text or in encrypted format.

If you enter a password or PIN in clear text, the server reads it during the next startup, encrypts it and writes it to the configuration file. From now on, the password and PIN information is no longer readable. If you are in doubt that the right password or PIN is set or if you need to set a new password or PIN, simply replace the encrypted value with the clear text value. During the next server startup, the password or PIN value is encrypted again.

In the *install_path*/**ssl/password.properties** file, the following parameters are available:

- **pin** - the PIN for the current private key for decryption of attributes (the default is **1234**). Is required if encryption mode is enabled.
- **previousPin** (optional) - the PIN for the previous private key for decryption of attributes. This allows smooth transition during key exchange / upgrade. The server is able to handle both old encrypted values (encrypted with the previous key) and new encrypted values (encrypted with the current key).
- **keystore** (optional) - the password for the SSL key store.
- **truststore** (optional) - the password for the SSL trust store.

In the *install_path*/**ids-j-***domain*-**S***n*/**private/password.properties** file, the following parameters are available:

- **domain** - the password for the user account which is used to access the configuration information in the LDAP directory.



The default user entry for the Connectivity domain is **cn=DomainAdmin,cn=*domain***,dxmC=Users,dxmC=DirXmetahub*. You can change it in the file:

*install_path*/**ids-j-***domain*-**S***n*/**bin/bindcredentials.xml**



The default user entry for the Provisioning domain is *cn=DomainAdmin,cn=**domain*. You can change it in the file:

*install_path*/**ids-j-***domain*-**S***n*/**bindprofiles/private/domain.xml**

- **signaturePin** (optional) - the PIN that is necessary for system client signature. The related certificate must be present in the Connectivity configuration at the DomainAdmin account under the Users tree (only visible in the Data View).



To leave a password empty, comment out the line with the hash tag (**#**) character. For example:
**`#previousPin=`**

#### 6.2.2.3. Starting the Java-based Server in Suspended Mode

You can start the Java-based Server in suspended mode. This mode avoids immediate action (for example, workflow starts) directly after startup. Note: Using this mode is only possible if you start the server from a command line. You cannot use this mode if you start the server as service.

After startup of the server, you can control the components via the Web Admin interface.

You can define the startup parameters either directly in the command line of the **runServer.bat** (or **.sh**) file, or you can define the parameters in an extra file. In this case, you must define this file in the relevant command line of the **runServer.bat** (or .**sh**) file:

**"%java_exe%"** ..... **-cfg config.cfg**

Then you must provide the parameters in this file:

*install_path*/**ids-j-***domain*-**S***n*/**bin/config.cfg**

These options are available:

- **server.suspend=true**
  Starts the server but stays in suspend mode. Use this mode if you need only monitoring access to the C++-based Server.
- **extension.load=-all**
  Prohibits loading configuration extensions; for example, to handle request or approval workflows.

  You can define specific extensions not to be loaded. These extensions are available:

  **com.siemens.idm.requestworkflow**
  **com.siemens.idm.realtimeworkflow**
  **com.siemens.idm.domcfg**
  **com.siemens.idm.backup**

  If you define **extensions.load=-com.siemens.idm.realtimeworkflow** then the real-time workflow engine is not loaded.

  **adaptor.load=***option*
  Lets you define which adaptors will be active after startup.
  These options are available:

**-all** - disables all adaptors
**+all*** - enables all adaptors
*-adaptor* - disables the adaptor specified in *adaptor*
*+adaptor* - enables the adaptor specified in *adaptor*

Example:
`adaptor.load=-all +DeadLetterQueue +EntryChangeListener +MailListener`

Disables all adaptors and enables only the internal event listeners (no external events are processed).

- **adaptor.suspend=***option*
  Allows suspending all or parts of the available adaptors after startup.
  These options are available:

  **\+all** - suspends all adaptors
  **-all** - enables all adaptors
  *-adaptor* - enables the adaptor specified in *adaptor*
  *+adaptor* - disables the adaptor specified in *adaptor*

  Example:
  `adaptor.suspend=-all +DeadLetterQueue`

  Enables all adaptors besides the dead letter queue.

Do not forget to reset the parameters in the **runServer** file; otherwise, the server will always start in this mode.

### 6.2.3. Recovery

Standard operating system features are used to implement recovery (watchdog functionality) on the supported platforms.

- On Windows, the standard recovery features for services are used. See the **Recovery** tab of the corresponding service.
- Due to missing standard features, there is no watchdog mechanism available on Linux.

### 6.2.4. Auditing

The Java-based Server provides a consistent mechanism to handle audit logs produced by workflow activities. Audit is a type of long-lived history data. It allows an auditor to review business events. Audit log entries are self-explanatory. They carry all data that is necessary to understand the event and the result.

Samples of auditable events are:

- A password has been modified successfully or the modification failed.
- An approval workflow has been started.
- Someone has approved a user-privilege assignment.

Read more about the format of the produced audit messages in the section "How Audit Trail Works" in the *DirX Identity Provisioning Administration Guide*.

The Java-based Server supports two audit handlers. Only one of them should be active at a time:

- The file-based audit handler.
- The JMS-based audit handler. If this one must be used, it first requires some manual installation steps. See the *DirX Identity Installation Guide* for details.

For configuring these audit handlers see the context sensitive help of DirX Identity Manager.

### 6.2.5. Statistics

The Java-based Server provides the following kind of statistical data:

- Tables of counters held in server components or produced by workflows. They can be viewed by the Web administration of the server (Web Admin).
- The statistics of each workflow stored in the LDAP configuration database. You can view this information with the DirX Identity Manager's Monitor View.

Read more about the statistics feature of the Web Admin interface in the section "Using Web Admin" in the *DirX Identity User Interfaces Guide*.

DirX Identity collects another set of statistics data and stores them in the LDAP Connectivity configuration. In DirX Identity's Connectivity view, select the Monitor View and open the **Event based** folder. You will find an entry for each workflow run. It contains the start and end time of the workflow, a table of statistical counters and a details list in the remark field. The statistic counters comprise the operations add, modify, delete and search and their result. The list below shows a brief summary for each request, containing the user whose password required changing and the result. This information is deduced from the audit log of each workflow. It is only visible if auditing for a specific workflow is enabled.

### 6.2.6. Logging

When the Java-based Server starts for the first time, default log levels are applied. They ensure that error and warning logs of all components are written.

The log files are stored in the **logs** folder of the server's home directory. You can configure how many records are written in one log file. Log file names start with **server*** and contain the timestamp.

You can view the log files from the file system using a standard editor such as Notepad or you can use the Web Admin tool. From the main menu, select **View log files** in the **Logging** section. The system presents the current log files.

To set and change log levels, use the Web Admin tool.

Log levels are specified for Java classes or packages. The system supports you by

presenting a list of logical components (server, several adaptors, connectors, ...) from which to choose. You can add class or package names by yourself in additional lines to specify your individual range of components to log.

## 6.2.7. Naming Schemes

Using multiple Java-based Servers for multiple domains requires an easy-to-understand naming scheme for service names, LDAP entry configuration names and file folder names.

#### 6.2.7.1. Java-based Server Object Naming

The Java-based Server objects in the Connectivity view group (Connectivity → DirX Identity Servers → Java-based Servers) follow this naming scheme:

*domain*-**S***n*-*hostname*

where

**domain**
is the domain for which this server is running.

**Sn**
is the server number (counts per domain).

**hostname**
is the host name where this server is running.

Example for a server object:

**My-Company-S1-myhost**

#### 6.2.7.2. Service Naming

The services on Windows use this naming scheme:

**DirX Identity IdS-J-***domain*-**S***n version*

where

**domain**
is the domain this server is running for.

**Sn**
is the server number (counts per domain).

**version**
is the version.

Example for a server:

```
DirX Identity IdS-J-My-Company-S1 V8.3
```

#### 6.2.7.3. File Folder Naming

The file folders in the installation area use this naming scheme:

**ids-j-***domain***-S***n*

where

**domain**
is the domain for which this server is running.

**Sn**
is the server number (counts per domain).

Example:

```
ids-j-My-Company-S1
```

Note that there is a folder **ids-j.org** which is a template used for creating new server connector frameworks. Do not change this folder and its content!

### 6.2.8. Resource Families

Use resource families to control the number of threads within a Java-based Server.

Each activity of a real-time or a request workflow is associated with a resource family: it requires that resource family. Java-based Servers provide resource families. An activity can only be processed on servers that host the required resource family.

Therefore, make sure you assign each relevant resource family to all of your Java-based Servers.

For each Java-based Server you must configure the number of threads per resource family. This allows you to influence to some extent the load distribution of certain workflow types between Java-based Servers: the slower a Java-based Server processes messages the fewer messages it will receive.

Use DirX Identity Manager to assign resource families and threads to a Java-based Server: select the server configuration entry in the Connectivity view (**Configuration** → **DirX Identity Servers** → **Java Servers -** *domain*), click the **Resource Families** tab, select the active resource families and then set the number of threads (two by default). Restart the server to make the changes effective. You can use Web Admin to check the configured number of threads.

#### 6.2.8.1. Understanding the Pre-Configured Resource Families

DirX Identity comes with a set of pre-configured resource families. You can use these resource families, add additional ones or exchange them completely with your set of resource families. To keep it simple, we recommend using the default resource families and then extending them as needed.

System-specific resource families (fixed values, not customizable) include:

**scheduler** - the internal scheduler of the server, which handles timeout situations and triggers retry of activities.

**workflowengine** - this resource family is reserved for the workflow engine itself, which starts and controls workflow activities.

**workflowscheduler** - the scheduler for real-time workflow schedules.

For each target system type, there is one default policy, for example ADS, LDAP, JDBC, Notes, SPMLv1, and so on.

Some others are for request and maintenance workflows:

**Apply** - default thread for request workflow activities to run processes that instantiate, modify and delete object changes.

**Calculate** - default thread for request workflow activities to run processes that calculate something (for example a GUID).

**Event_Maintenance** - default thread to handle event-based processing activities.

**Mail** - default thread to handle activities that send mail requests (for example all error or notification activities).

**Request_Workflow** - default thread to handle Java-based join activities of provisioning workflows that provide manual provisioning via request workflows.

### 6.2.9. JMX Access to the Java-based Server

By default, JMX access to the Java-based Server needs password authentication.Authentication is performed by passing the user credentials to an LdapLoginModule which tries to bind with these credentials to the Connectivity LDAP server.When the bind is successful, the JMX access is successfully authenticated.

Per configuration, the LDAP user is **cn=DomainAdmin,cn=*** ***domain***,dxmC=Users,dxmC=DirXmetahub*.

On the JMX client side, you only need to give the “DomainAdmin” as username and the appropriate password.This is configured in the file **jmxldap.cfg** in the **bin** folder of the Java-based Server.

If the SSL flag is activated in the system-wide configuration, JMX access is also secured by SSL.In this case, non-SSL access is not possible.

If SSL is configured for the Connectivity store, the authentication process to the LDAP server is also performed using an SSL connection.In this case, you need to make sure that the LDAP server’s root CA chain certificates are in the **cacerts** file of the Identity Java environment.

# 6.3. Managing the C++-based Server

This section describes how to manage the C++-based Identity Server (IdS-C), including how to:

- Install, start, and configure C++-based server components

## 6.3.1. Server Components

The C++-based Identity Server (IdS-C) consists of a set of services that represent the required functionality.The main components start Tcl-based workflows according schedules and control their activities.A status tracker runs on exactly one IdS-C Server for which the **dxmRunStatusTracker** attribute is true.This status tracker is responsible for updating the status of Tcl-based workflows in the Monitor area of the Connectivity database.

Each server is represented by a service on Windows or a daemon process on UNIX.

On Windows:

- **DirX Identity IdS-C** *version*

On UNIX:

- **DirX Identity IdS-C** *version*

DirX Identity IdS-C *version* must run on each machine on which you have performed a C++-based Server installation.

All DirX Identity components communicate with each other using TCP/IP based protocols:

- Data is read from and written to the configuration database in the LDAP directory (by default, port 389)
- Messages are transferred between components via the Messaging service.

#### 6.3.1.1. Starting Up the Server Components

The services that make up the C++-based Server normally start automatically when you boot your system:

The DirX Identity IdS-C service starts independently from the DirX Server service.

- Service name on Windows: **DirX Identity IdS-C** *version*
- Process name on Windows: **dxmsvr.exe**
- Process name on UNIX: **dxmsvr**
- Controlled processes on Windows: **dxmmsssvr.exe**
- Controlled processes on UNIX: **dxmmsssvr**

The shell script *install_path*/etc/S99dmsvr* starts the DirX Identity IdS-C service on Linux

when the system runs in multi user mode.

The service uses the following configurable polling mechanism during start-up:

- Try to connect to the LDAP server. If this fails, try again the number of specified steps after the specified time in the initialization file of the C++-based Server.
- If there is no access to the LDAP server, the service start is aborted.
- When the bind to the LDAP server is successful, the necessary information is read from the Connectivity configuration database (object Messaging Service in the Expert View: Configuration → Messaging Services).

If the DirX Identity IdS-C Service detects that a C-based Server crashed, the service restarts the server. Permanent threads (Status Tracker and Scheduler) running in the C-based Server are also checked regularly by the **Keep Alive** mechanism and restarted automatically if they are not responding. You can check the threads with the Get Server State command for each individual server.

You can use the following parameters in the **[settings]** section of the C-based Server initialization file (dxmmsssvr.ini) to define the polling parameters for each machine that runs a C-based Server:

- **timeout** - the time between two polling cycles (the default is 45 seconds).
- **repeat** - the number of retries to perform (the default is 10 times).

By default, DirX Identity tries to connect 450 seconds to the LDAP server.

#### 6.3.1.2. Configuring the C++-based Server

The C++-based Server is controlled by the initialization file **dxmmsssvr.ini**, which is located in the *install_path***/server/conf** subdirectory.

The parameters in the **[settings]** section are necessary for the registration of the server in the connectivity configuration database and for consistency check during startup:

- **dnServerName:** the name of the C++-based Server's configuration object in the connectivity configuration in the folder **DirXmetahub Servers** (default name: **main**)
- **host:** the name of the server on which this C++-based Server instance runs (for example: abc123.myCompany.de or 123.54.76.11).

These two parameters are used to verify that the system is consistent (the right server registers to the correct server LDAP entry). If you change these parameters by hand, you must also change the corresponding parameters in the configuration database. The check is defined as:

The field **dnServerName** in dxmmsssvr.ini file is equal to the attribute **dxmDisplayName** of the C++-based Server object (the displayed name) and
the field **host** in dxmmsssvr.ini file is equal to the attribute **dxmServerName** of the service object to which the C++-based Server object points.

For the host field check, examine the link to the Service object of the relevant C++-based

Server object. Please note that relevant for this field is the **Server Name** field in the Service object, not the **IP Address** field.

If this check fails, the next start of the C++-based Server will fail! Correct either the LDAP object or the **dxmmsssvr.ini** file.

- **cconnserver** - whether the connector server is started (default is **0**):
  **0** - Connector server will not be started (no connectors are running).
  **1** - Connector server is started (configured, active connectors are running).
- **timeout** - the time between two polling cycles to connect to the LDAP server (the default is 45 seconds).
- **repeat** - the number of retries to connect to the LDAP server (the default is 10 times).
- **encryptionmode** - whether the server should use data encryption or not (the default is 0, which means no data encryption). Set this parameter to 1 if the server should use data encryption. In this case set the **pin** parameter, too.
- **server_restart** - the number of server restarts the watchdog performs during startup.
- **IgnoreSaveStatusInfoError -** whether or not errors in saving activity status info results are ignored (the default is **0**, which means that errors are not ignored). If set to **1**, errors from sending activity status info are ignored and the normal workflow execution continues.

The parameters in the **[metadir]** section are the parameters for the LDAP bind to the configuration directory:

- **server** - the name of the directory server (for example: abc123.myCompany.de).
- **user** - the distinguished name of the account which is used by the C++-based Server to access the LDAP directory (for example: cn=server_admin, dxmC=dirXmetahub).
- **pwd** - the password for this user account. See the section Password Handling for more information how to set passwords.
- **port** - the LDAP server port (the default is **389** for non-SSL access).
- **ssl** - whether or not SSL access to the LDAP server will be used (**ssl=1**). The default value is **ssl=0**. See the server SSL connections topic for more information.
- **cert-db-path** - the path to the **cert8.db** file (only used when **ssl** is set to true). The default setting is empty.



The PIN for the current private key for decryption of technical bind passwords and attributes (and the previous PIN) and the key store password (for the SOAP secure key store) are maintained in the file *install_path*/ssl/password.properties*.

#### 6.3.1.3. Changing the Service Login Account (Windows)

On Windows only, you can change the account that the service uses to log on.To change it, run the Initial Configuration with the C++-based Server step again and change the account.

#### 6.3.1.4. Server Password Handling

During startup, all DirX Identity servers require reading the relevant configuration information from the Identity Store.For authentication, passwords and PINs must be present in the password configuration files.The servers can read passwords or PINs in clear text or in encrypted format.

If you enter a password or PIN in clear text, the server reads it during the next startup, encrypts it and writes it to the configuration file.From now on the password and PIN information is no longer readable.If you are in doubt that the right password or PIN is set or if you need to set a new password or PIN, simply replace the encrypted value by the clear text value.During the next server startup, the password or PIN value is encrypted again.

## 6.4. Distributed Deployments and Scalability

This section describes the powerful capabilities of DirX Identity for setting up a highly distributed environment and supporting scalability.

Components that can be distributed include:

- The LDAP server(s) with the DirX Identity domains and the Connectivity Database.
- One or many Java-based Servers.Several such servers can be installed on the same host system - even several associated with the same Identity domain.
- Message Broker instances on multiple servers.
- One or many C++-based Servers.At most one such server can be installed on a single host system.
- Other components, such as Business User Interface, Web Center, Identity Manager and the Web Services (both SOAP and REST).These components can be distributed freely.

Except for the communication protocols to the target systems, the required communication protocols are:

- LDAP for access to the LDAP server(s).
- TCP/IP for communication from and to the Message Brokers.
- SOAP/HTTP to access the request workflow service.
- SOAP/HTTP to the SPML Provisioning Service.
- HTTP to the REST service.
- JMX for the supervisors and Server Admin / Web Admin.

The configuration of the distributed server deployment is stored in the LDAP server with the Connectivity Database. The DirX Identity components Manager and Server Admin give an overview and allow changing these settings.

In addition to server and system information, the Connectivity Database stores configuration data on maintenance and Provisioning workflows and their schedules as well as their monitoring data.

While multiple DirX Identity domains can be stored in the same LDAP server, this is not possible with multiple Connectivity Databases: There can be at most one in a LDAP server. This results in the following deployment options for multiple DirX Identity domains:

- Each domain is associated with its own Connectivity Database. As a result, you need one LDAP server per domain, which also contains the Connectivity Database.
- Several domains are associated with the same Connectivity Database, which allows you to store all in the same LDAP server.

The next sections present different typical deployments for DirX Identity and their corresponding strengths and weaknesses, including:

- "All-on-one-machine" deployment
- Distributed deployment of Java-based Servers with Java-based provisioning and request workflows
- Distributed deployment of C++-based Servers with Tcl-based workflows and issues with using a shared file system in distributed deployments.

### 6.4.1. All on One Machine

The deployment for a DirX Identity domain requires as a minimum the following components:

- The Connectivity Database.
- The DirX Identity domain.
- One Java-based Server with the Request Workflow Timeout Check, all adaptors activated and the scheduler for the Java-based workflows.
- One C++-based Server controlling all Tcl-based workflows and the status tracker.
- One Message Broker.

Note that one Java-based Server is associated with one DirX Identity domain and cannot serve multiple domains. But one C++-based Server can run Tcl-based workflows for several DirX Identity domains.

In the simplest installation, all of these components can be deployed to the same host system. This variant is the easiest to set up and maintain.

### 6.4.2. Distributing Java-Based Servers

As a Java-based Server can serve only one domain, you need at least as many Java-based Servers as domains: at least one per domain.

You can also have multiple Java-based Servers for one domain. This configuration helps you to distribute load: request, provisioning and event-based workflows.

Read the section about naming schemes to understand how services, LDAP configuration objects and file folders are structured.

#### 6.4.2.1. Distribution Criteria

A Java-based Server can run several types of workflows: event-driven and scheduled workflows for provisioning, event-driven workflows for entry changes, scheduled workflows for maintenance and request workflows for approvals.

All of them can run on every server and all of them can run in parallel on more than one server.

To run workflows of a certain type on a Java-based Server, you must activate the corresponding adaptors. A workflow is only started via a message. The messages contain the events to trigger, for example, the provisioning of an account or group and the requests to start a workflow by schedule or manually from Identity Manager. JMS adaptors read the messages from the specific queues of the Message Broker and start the appropriate workflow.

Here is a list of workflow types, associated adaptors and queues. For a short description of the queues, see the section “Managing the Message Broker” → “Using Messages in DirX Identity” → “Queues of the Java-Based Server” in this guide:

| **Workflow Type** | **Adaptor** | **Queue (without the domain prefix)** |
|---|---|---|
| *Provisioning* | Provisioning Request Listener | dxm.request.provisiontots._default |
| | Provisioning Request Start Workflow Listener | dxm.request.workflow.provisiontots._default |
| | Import to Identity Listener | dxm.request.importtoidentity._default |
| *Password Changes* | Password Change Listener | dxm.event.pwd.changed |
| | Account Password Change Listener | dxm.event.SvcTSAccount.pwd |
| | Set Account Password Listener | dxm.setPasswordRequest._default |
| *Request Workflows* | Request Workflow WorkflowEngine Listener | dxm.requestworkflow.workflowengine |
| | Request ActivityTask Listener | dxm.request.activityTask |
| *Mail* | Mail Listener | dxm.notify.mail |
| | Text Message Listener | dxm.notify.sms |
| *Maintenance* | Entry Change Listener | dxm.event.ebr |
| | Entry Change Start Workflow Listener | dxm.request.workflow.ebr |

You can also run workflows for a selected connected system on dedicated servers. For details on how this works, see the section “Separating Traffic for Selected Connected

Systems”.

Some components must run on exactly one server per domain; these are the Java scheduler and the Request Workflow Timeout Check (also called FullCheck).

The Java scheduler starts workflows by sending a start message to the Message Broker. There are two special adaptors responsible for consumption of these messages: the Entry Change Start Workflow Listener and the Provisioning Request Start Workflow Listener. The Entry Change Start Workflow Listener needs to be co-located with the Entry Change Listener; the Provisioning Request Start Workflow Listener needs to be co-located with the Provisioning Request Listener.

Within a Java-based Server, resource families control how many threads a workflow - more precisely, an activity within a workflow - can use. To some extent, this also influences the number of events the server processes. The more threads you reserve for a resource family, the more events or workflows can be processed by that server and the more events it obtains from the Messaging service.

Several Java-based Servers - of the same domain or of different ones - can run on the same physical system. The number of CPUs determines whether this makes sense: the more CPUs there are, the more threads can run in parallel.

#### 6.4.2.2. Configuring One Java-based Server per Domain

The simplest deployment is to run one Java-based Server per domain. This server then runs the Java scheduler, all real-time and all request workflows.

When one configuration database covers several provisioning domains, then you must set the flag **Include domain in topic** at the corresponding domain object.

This setting forces the JMS clients to include the domain name into the queue and topic names. Especially JMS adaptors in a Java-based Server only read from queues whose names start with their domain name.

In the Connectivity Database, follow the wizards and store the schedules, workflows, jobs and connected directories in domain-specific folders. A Java-based Server only loads workflows from its associated domain folder.

#### 6.4.2.3. Configuring Multiple Java Servers per Domain

To set up multiple Java-based Servers per domain:

- Run the configurator (either **Configuration** or **Initial Configuration**).
- Define the domain in the **Domain Configuration** step
- Select **Create a new Java Server** from the drop-down list of the **Server to update or create** field of the **Java-based Server** step.
- Define the relevant parameters. Make sure you use free ports.
- Repeat this step for additional Java-based Servers you want to create.
- Start DirX Identity Manager (**Connectivity** view group) and select **Expert View**.

- Open **Configuration** → **DirX Identity Servers** → **Java Servers** → *domain*. You should see all your configured server instances and if you open them all movable adaptors.
- Select **Manage IdS-J Configuration** from the context menu of a Java-based Server node.
- Assign the movable adaptors, the scheduler and the request workflow processing to the servers. Deactivate unused adaptors.
- Click **OK** to store the configuration or **Cancel** to abort it.
- Restart all Java-based Servers to load the changed configuration.
- Use Server Admin to check that the adaptors are configured correctly.

### 6.4.3. Separating Traffic for Selected Connected Systems

You can separate the traffic for synchronizing selected connected systems completely from the synchronization of other connected systems. It is possible to dedicate the provisioning of a target system to one or more Java-based Servers and to separate threads within a Java-based Server.

Reasons for such a configuration can include:

- Separating the traffic for target systems with many events from others
- Separating slow target systems from others
- Running workflows for a specific target system behind the firewall
- Support for file-based workflows where the files to import or export are only accessible from a dedicated system
- Always running workflows for one target system on the same server for easier problem analysis
- Support for the DirX Audit History synchronization workflows that are not associated with any target system

Within a Java-based Server, the threads for processing workflows for such selected connected systems are decoupled from those processing other connected systems. So even if you have only one Java-based Server deployed, the provisioning of a slow target system does not slow down the provisioning of the other target systems.

Note that this feature can also be applied to connected directories that are not associated with any target system. Prominent examples are source systems from where entries, namely users are imported to the Identity Store. Another example is the DirX Audit History Database with its synchronization workflows.

To assign a target system or a cluster of target systems to a Java-based Server, assign the corresponding connected directory (a cluster has only one connected directory) to the server(s):

- In DirX Identity Manager's Connectivity view group, select **Expert View**.
- In the folder **Connected Directories**, select the connected directory. Open the tab **Connected Directory**.

- In the section **Associated Server**, select one or more Java-based Servers.
- If you want to have more than one thread processing provisioning requests for this connected system, enter the number in the optional **Listeners per Target System**. The default is 1. Note: this number applies only to the queues **dxm.request.provisiontots** and **dxm.request.importtoidentity** (see below). For processing of password changes and for running a complete validation or delta synchronization, one thread per queue should be enough. As an exception, if the connected directory has no associated target system, the number of threads is applied to the queue **dxm.request.workflow.provisioning**. In this special case, it is possible to run multiple full or delta synchronization workflows in parallel.

To activate this configuration, either:

- Restart the servers or
- Select each server in **Configuration** → **DirX Identity Servers** → **Java Servers** and then select **Load IdS-J Configuration** from the context menu.

The servers use the following queues:

*domain*.**dxm.request.provisiontots.***target system identifier*

*domain*.**dxm.request.workflow.provisioning.***target system identifier*

*domain*.**dxm.setpasswordrequest.***target system identifier*

*domain*.**dxm.request.importtoidentity.***target system identifier*

If there are target systems associated with the connected directory, the *target system identifier* is built using the attributes type, cluster and domain of the target system in lowercase: *type**.***cluster***.***domain*****. For a target system that is part of a cluster, the *domain* part is empty and the target identifier is built as *type*.*cluster*. For example, an Active Directory with a forest name "Europe" and domain "Germany" has the identifier *ads.europe.germany**. If this target system is part of a cluster, the identifier is **ads.europe**.

If there are no target systems associated with the connected directory:

- Messages for starting a workflow providing its DN go to the queue *domain**.dxm.request.workflow.provisioning* and the target system identifier is built using the type and the display name of the connected directory in lowercase: *type* *.***displayname*****. For example, a file with the display name "LDIFfile" has the identifier *ldif.ldiffile**.
- For messages to import a given entry to Identity Store, the queue name *domain**.dxm.request.importtoidentity* is extended with the **whenApplicable** section of the workflow identifier. For example, a workflow with type "ldap", cluster "import" and domain "userldap" has the identifier **ldap.import.userldap**. This allows running workflows to import users and business objects from the same source directory in separate threads.

Please avoid using special characters in type, cluster, and domain names as they are used to build queue names dynamically. Especially avoid using the following characters: ".", "*", ">", "?", "\", "/".

The adaptors for the queues *domain**.dxm.request.provisiontots*, *domain**.dxm.request.workflow.provisioning*, *domain**.dxm.setpasswordrequest*, and *domain**.dxm.request.importtoidentity* dispatch the messages to either the appropriate target system specific queue or to the respective default queue (for example, *domain**.dxm.request.provisiontots._default*). For information about which servers process the default queue messages, see the section “Distribution Criteria”.

These dispatchers and the adaptors for the target system-specific queues follow a behavior that is different from those for the normal provisioning queues:

- They do not store the received messages in their own file repository and therefore need no special handling for high availability. Instead, the adaptors process each message immediately and acknowledge it to the Message Broker only when processing is finished. In case of breakdown, the not-yet-acknowledged messages are still available in the Message Broker and are delivered when the adaptor re-connects.
- They do not use the workflow engine. Instead, they perform the error handling on their own. They pass the messages directly to the join activity of the workflow. If an error occurs, they send the message again to the Broker with a delay according the configured retry wait time. If that is not possible because the error is not considered temporary or the retry limit is reached, the adaptor runs the error activity and sends the message to the Dead Letter Queue.

You can monitor the dispatchers and adaptors both in WebAdmin and with tools like Nagios using JMX MBeans.

With WebAdmin, select the following items in the left-hand tree:

- Provision Dispatchers - in the details view at the right, you’ll see a table with the queues the dispatchers listen to and the number of messages received and failed; that is, those that could not be forwarded to the target queues.
- Provision TS Listeners - in the details view at the right, you’ll see a table with the queues target system-specific adaptors are listening to and the number of messages that have been received and processed either successfully or not. Note that re-delivered messages in case of temporary errors are only delivered to the same target system-specific queue and not again to the queue on which the dispatcher is listening. So, they are visible only in this view.

The object names of the JMX MBeans for dispatchers and target system-specific adaptors start with **com.siemens.idm:type=idsj** and then identify the listener and the queue by the parameters “topic” and “name”:

- Dispatchers - **com.siemens.idm:type=idsj,topic=ProvMsgDispatcher,name=****queue***, where values for *queue* are *provisiontots**, **setpasswordrequest**, **importtoidentity** and **workflow.provisiontots**.
- Target system-specific adaptors - **com.siemens.idm:type=idsj,topic=ProvTSListener,name=****queue*.*target-system* **where *queue* names are the same as for the dispatchers and *target-system* is in the format *type*.*cluster***.**resource*.

### 6.4.4. Distributing C++-based Server Components

In all configurations, we assume that we are handling a two-step Tcl-based workflow (a workflow with two activities) that exports data from the Identity store and imports data into Active Directory. Other types of workflows with only one activity (for example, an LDAP2LDAP workflow) or with more than two activities can be discussed in a similar way.



We assume that all computers that belong to the DirX Identity domain must be time synchronized using operating system mechanisms. Otherwise, scheduling conflicts or incorrect runs of a distributed workflow can occur.

For the two-step workflow, the following typical configurations exist:

- All parts on central server
- Target activity distributed
- All parts on target server

As a minimum for running Tcl-based workflows, the following components are required:

- One C++-based Server (IdS-C). It contains the status tracker and a scheduler for Tcl-based workflows.
- One Message Broker.
- The LDAP server storing the Connectivity Database and the identity domain (users, accounts, groups, and so on) to be provisioned.

For understanding the discussions below, you should be aware of the following:

- A Tcl-based workflow is associated with a system, thus with the IdS-C server running on this system. The scheduler and the workflow engine in this server are responsible for processing and controlling the workflow.
- A target activity is associated with a system and with the IdS-C server on that system. This IdS-C server is responsible for processing and controlling this activity.

The following sections discuss recovery and safety mechanisms as well as security issues of the different alternatives.

#### 6.4.4.1. All Items on Central Server

In general, it makes sense to keep all components on the same machine to optimize performance and minimize the dependency from network malfunction.

In this configuration, both the C++- and the Java-based Server together with the messaging service, the workflow and all activities run on the central server.

The IdS-C server reads workflow configuration data from the Connectivity Database. In the first activity, **metacp** reads data from the Identity store via LDAP and stores the processed result to a file. In the second activity, the ADS agent reads this file and writes to Active Directory on the target machine via the native interface. The following figure illustrates this

configuration.

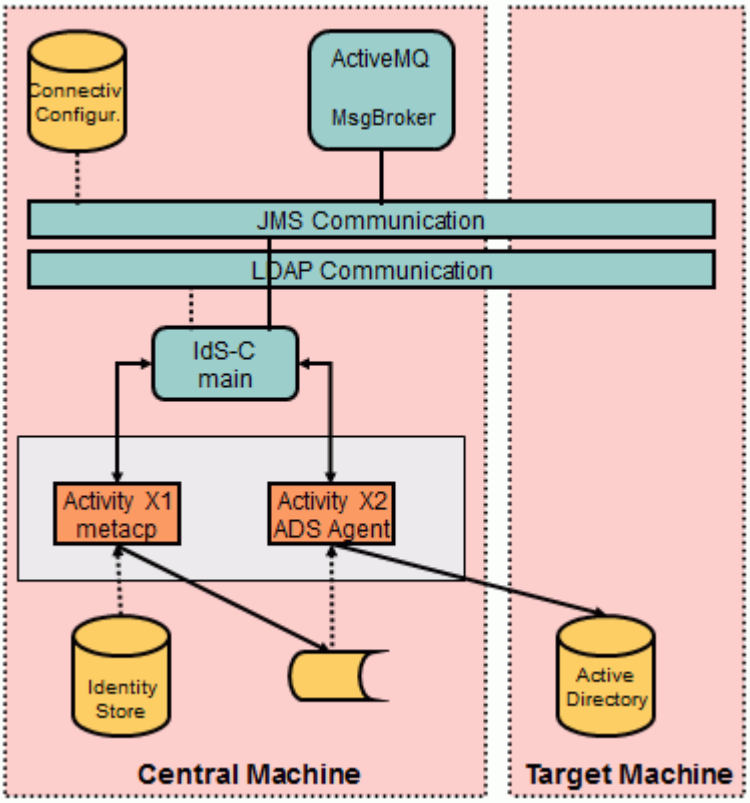



*Figure 22. "All Items on a Central Server" Configuration*

**Strengths:**

- All DirX Identity processes run on the central machine, making administration easy.
- The data file does not need to be transferred via the network. It is only visible on the central machine. No shared file system needs to be set up and maintained.
- The target server (hosting Active Directory) does not require the installation of any DirX Identity component.
- All communication except for the one to the target system API depends only on a running central machine (especially all JMS messages can always be sent and delivered). The availability of this machine defines the availability of the entire DirX Identity domain.

**Weaknesses:**

- If all workflows run on the central machine, heavy load could be the result. Be careful to distribute the schedules accordingly to avoid this situation.
- Data is transferred via the native target system API (in our example, the Active Directory API). Security is determined by the features of this interface.
- You must ensure that the network connection is fast enough for the specific agent.

**Recovery:**

For the basic recovery features, see the section "Recovery and Safety Mechanisms".

If the central machine breaks down and is re-booted during a workflow run, it cannot be guaranteed that the workflow will be restarted.

#### 6.4.4.2. Target Activity Is Distributed

In this configuration, the Identity Servers with the messaging service, the workflow and the first activity (**metacp**) run on the central machine.

**metacp** reads data from the Identity store via LDAP and writes the mapping result to a file. The file must be transferred via the network or made accessible through a shared file system. The ADS agent then reads it and imports the entries to Active Directory on the target machine via the native interface. The following figure illustrates this configuration.

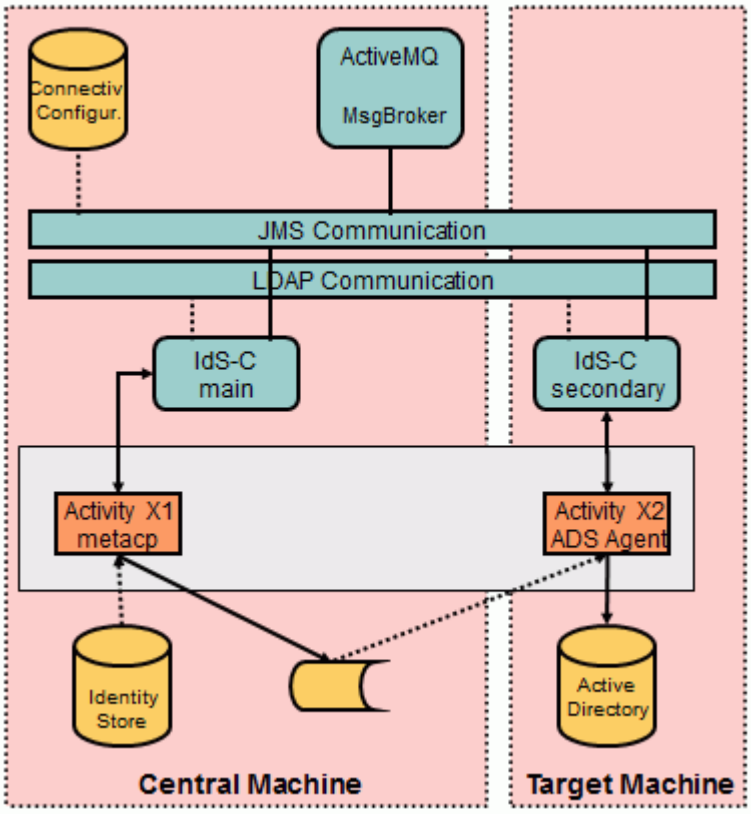



*Figure 23. "Target Activity is Distributed" Configuration*

**Strengths:**

- The target system interface is accessed on the target machine, so it is not visible on the network.
- You can distribute load over the network.
- The configuration information that is transferred via the LDAP connection between the different C++-based Servers can be secured via SSL.
- The scheduler runs only on the central machine and can therefore send the relevant messages to start workflows to the messaging service. The messages are stored permanently and are resent until the deviation time is reached.

**Weaknesses:**

- A C++-based Server and the ADS agent must be installed and maintained on the target machine.
- The data file must be transferred via the network or made accessible via a shared file system. In both cases data is visible on the network. Use the encrypted file transfer to secure the file transfer over the messaging service.
- Shared file systems if used are an additional administrative task.
- If the network is not available, the C-based Server on the central machine cannot send messages to the C-based Server on the target machine to start the agent. The workflow will fail in this situation. It is started again at the next scheduled time or after a defined retry interval.
- If the network is not available, the C++-based Server at the target machine cannot deliver status information to the messaging service. This information is lost.

**Recovery:**

For the basic recovery features, see the section "Recovery and Safety Mechanisms".

If the central machine breaks down and is re-booted during a workflow run, it cannot be guaranteed that the workflow will be restarted.

#### 6.4.4.3. All Items on Target Server

In this configuration, the workflow and both activities run on the target machine, while the LDAP server and the messaging service remain on the central machine.

**metacp** reads data from the Identity store via LDAP over the network and writes the mapping result to a file. The ADS agent then reads this file and writes it to Active Directory on the same machine via the native interface. The following figure illustrates this configuration.

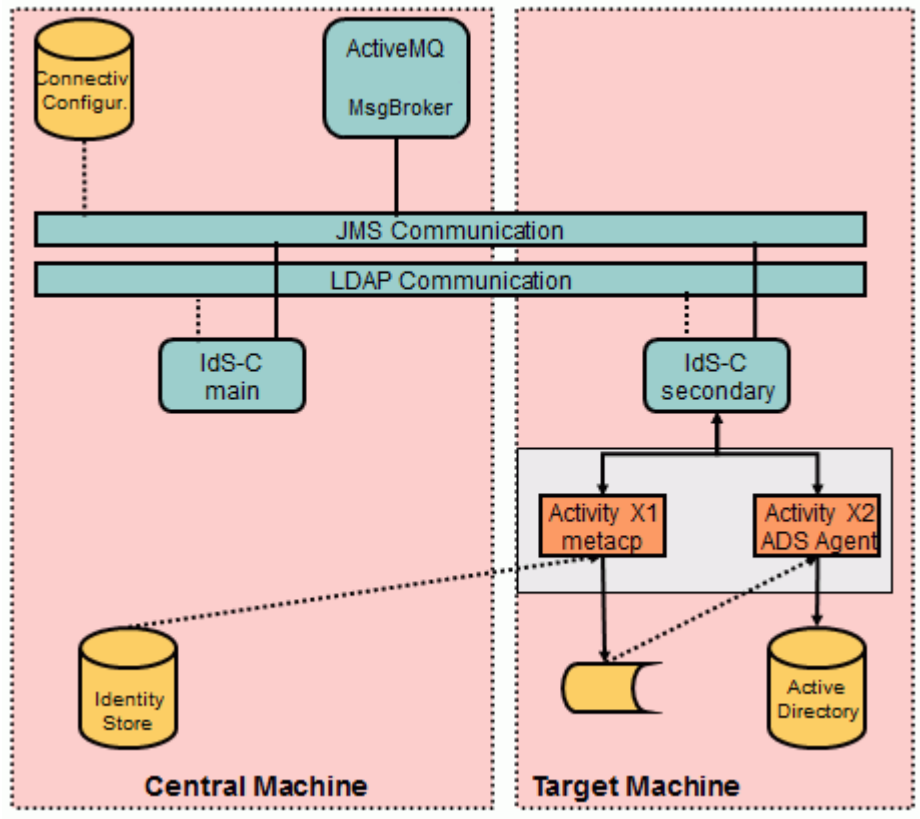



*Figure 24. "All Items on a Target Server" Configuration*

**Strengths:**

- The target system interface is accessed on the target machine, so it is not visible on the network.
- You can distribute load over the network.
- The data file does not need to be transferred via the network. It is only visible on the target machine. A shared file system does not need to be set up and maintained.

**Weaknesses:**

- A C++-based Server, **metacp** and the ADS agent must be installed and maintained on the target machine.
- If the network is not available, the C++-based Server on the target machine cannot send messages to the messaging service to start the agents. The workflow will fail in this situation. It is started again at the next scheduled time.
- If the network is not available, the C++-based Server on the target machine cannot deliver status information to the messaging service. This information is lost.

#### 6.4.4.4. Reduce File Handling in Status Area

In order to improve scalability, you should consider reducing status creation and file handling of Tcl-based workflows. Identity Manager allows you to configure this on a fine-grained basis:

- For each file, set the **Save Mode** and **Copy to Status Area** flags correctly to avoid

unnecessary saving of files, especially if the file size is large.

- Compressed status entries can reduce workload on the DirX Identity status tracker. Choose an appropriate detail level of the **Status Compression Mode** option at the central configuration object or at individual workflow objects. You can reduce the detail level or completely suppress status entries for workflows that succeeded.

#### 6.4.4.5. Setting Up a Shared File System for Distribution

You can run workflows in a distributed environment. By default, the file service automatically handles all necessary file transfers when the activities run on different machines. To enhance performance, you can set up shared file system information in the relevant tab of the DirX Identity server object.

Follow this general procedure:

- Select the correct C++-based Server for each workflow and activity. The best method for viewing and adjusting these parameters is to use the workflow structure view, which you can access from the Global View and Expert View.
- DirX Identity detects necessary file exchange between activities automatically if the connectivity is defined by channels.
- You cannot use the system account in a Windows environment to run your C++-based Servers (they are not allowed to access network resources).Use other accounts that can access network resources instead.

See also the section "Workflow Design Rules" for valuable hints about this subject.

### 6.4.5. Distributing Message Broker Instances

Multiple Message Broker instances can be spread over IdS-J and external servers.As only one of these Message Brokers is accessible by the clients, this setup focuses on high availability and not on scalability.

The important deployment configuration is related to the persistent message store, which must be located on a shared drive to which all Message Broker instances have access.

## 6.5. High Availability and Recovery

DirX Identity has some built-in mechanisms that avoid error situations or help to recover from them:

- The messaging service stores messages permanently in its messaging repository.
- The scheduler starts workflows at a defined time (at the latest during the deviation time after the workflow start time).Thus, you can be sure that your workflow does not run at times when it should not run.
- You can define a retry interval for each schedule.If a workflow fails and the deviation time is not over, DirX Identity restarts the workflow until it is okay.This feature can overcome temporary errors (for example, the network is temporarily not available).

- All messages except for status tracker messages have a **lifetime** that guarantees that actions are not started after the defined **timeout** of the workflow.The messaging service deletes these messages automatically when they have timed out.
- The status tracker messages have no **timeout** set.Thus, all status messages that the messaging service has saved are delivered when the network and the messaging service are available again.

During runtime, DirX Identity produces and manages data in various repositories, including:

- **Repository of Message Broker** - contains the messages that have not yet been consumed by Java server adaptors; for example, password changes or events for real-time provisioning.The repository should be located on a shared network device.
- **Repositories of adaptors in Java-based Servers** - contain the messages for which a real-time workflow has been triggered, the Dead Letter Queue and - if High Availability is enabled - also the backup of the monitored adaptors.
- **LDAP directory** - contains configuration and status data; for example, the current state of a request workflow or information about the last delta run of a Provisioning workflow.

Note that the adaptors for target system-specific messages as well as the Resolution Adapter don’t have a repository.They process each message immediately and therefore can acknowledge and thus delete it at the Message Broker automatically after it has been processed.

When high availability is enabled, the adaptor repositories of one Java-based Server are backed up in real-time from the backup adaptor in another, the monitoring Java-based Server.In case of a system crash, either the monitoring supervisor automatically or an administrator manually using Server Admin instructs the monitoring Java-based Server to restore the messages from the repository backups and send them to the message broker.

In addition, or as an alternative, you can perform a scheduled backup of the Java-based Server adaptor repositories; for example, every day.This helps you to restore the system (recovery) or to set up the system at another location (disaster recovery).For configuration of a joint backup workflow, see the section "Joint Backup Workflow" in the chapter "Maintenance Workflows" in the *DirX Identity Application Development Guide*.Currently you should set up a directory backup at the scheduled backup time by hand.

# 6.6. Diagnostic Information

The Java-based Server and the C++-based Server write valuable information during server startup.

The servers write a diagnostic file during each start.It consists of two sections:

System Information - provides information about the hardware and operating system in use.

Server Information - provides a set of server-specific properties and information.

This information is useful for determining the conditions of a specific server run; for example, the one where an error occurred.

For detailed information, see the *DirX Identity Troubleshooting Guide*.

## 6.7. Managing Daylight Savings Time

All parameters that define dates in schedules within DirX Identity are stored in GMT format, which means that a workflow runs at fixed world time.However, the display of these parameters is in local time to allow fpr consistent handling of workflow starts in a worldwide distributed DirX Identity scenario.

Working in a country that switches between summer and winter time (daylight savings time) may require you to adjust the schedules so that they start in relation to fixed local time.For this purpose, you can use the Tcl script **shiftSchedules.tcl** in the folder *install_path***/tools/scheduling**.

To adjust the timing:

- Customize the script's bind parameters before you use it, including the bind password.We recommend that you operate on a copy of the script and protect the script against unauthorized read/write access.
- If the clock has been moved forward one hour, adjust the schedules (-1 h) with the command:*
  metacp shiftSchedules.tcl backwards*
- If the clock has been set back by one hour, adjust the schedules (+1 h) with the command:
  **metacp shiftSchedules.tcl forwards**
- The script writes a log file **shiftSchedules.log** into the same folder in which **shiftSchedules.tcl** is located.
- You can also call the script with an absolute path, for instance:*
  metacp /home/metatest/myLocation/shiftSchedules.tcl forwards*
- Scripts that call this script must contain an initialization part that provides the correct runtime environment for executing **metacp**.For UNIX, this is done by a directive in the form *install_path***/.dirxmetarc**.
- Use the operating system scheduler to start this script regularly when winter or summer time begins.

## 6.8. Connector Frameworks

DirX Identity provides two types of connector framework that can be used to control additional custom target system APIs:

- Identity Connector Framework for Java - allows controlling event based synchronizations to target systems that provide Java interfaces.
- Identity Connector Framework for C/C++ - allows controlling event based

synchronizations to target systems that provide C or C++ interfaces.

The following sections provide a high-level overview of each framework.For detailed information, see the *DirX Identity Integration Framework Guide*.

### 6.8.1. Identity Connector Framework for Java

The Identity Connector Framework for Java provides a comprehensive set of functionality that is built completely on Service Provisioning Markup Language (SPML). The following figure illustrates its components.

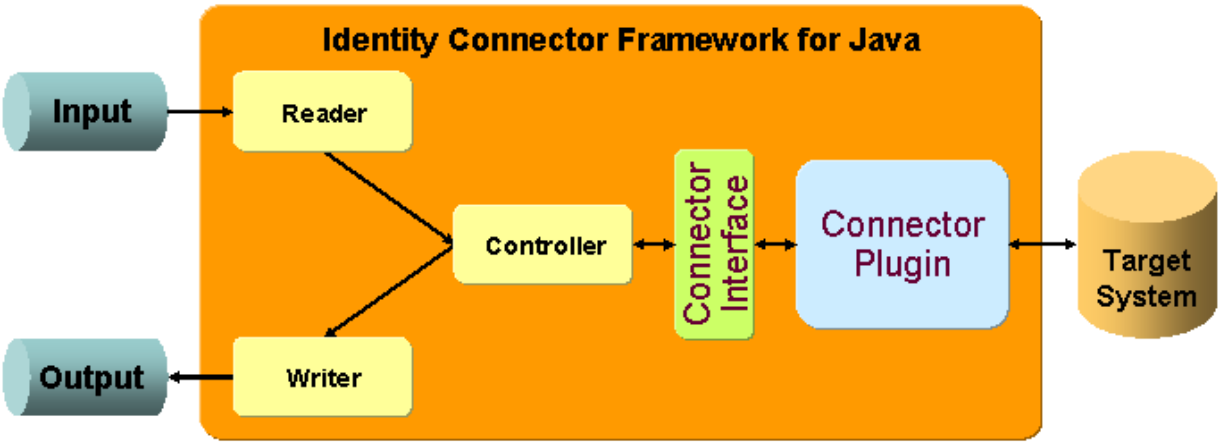



*Figure 25. Identity Connector Framework (Java) Components*

Its most important components are:

- **Reader** - allows for the transformation of external formats into SPML format (for example, LDIF change files)
- **Writer** - transforms internal SPML format to external formats (for example, LDIF content files)
- **Controller** - controls all other components
- **Connector Interface** - the only interface the Connector Plugin must recognize. Provides all information in SPML format.
- **Connector Plugin** - the piece of custom code that maps the SPML internal format to the API calls for a specific target system.

The following example explains how a connector could work (let's assume the connector is waiting permanently for password changes):

- After startup, the controller gives control to the reader.
- The reader either reads events from a channel in event-driven workflows or from a file in scheduled workflows. It transforms the data into SPML format; for example, a modify request to set a password.
- The controller takes the SPML request and passes it to the request transformer. This component knows how to map the attributes in the SPML request and generates a

modified SPML request. The request transformer is optional.

- The controller takes this request and passes it to the connector plugin. The connector performs the modification request with the delivered attributes.
- The connector returns an SPML response with the result of the operation.
- The controller passes this information to the optional response transformer, which maps the information accordingly and returns it to the controller.
- The controller takes it and informs the writer to send the response. In an event based workflow, the writer puts the response into the out channel of the activity. From there the workflow engine passes it to the adaptor. A SOAP adaptor would return this to the SOAP client. The message queue adaptor ignores it. In case of a batch like job, the response writer typically stores the response into a file.

### 6.8.2. Identity Connector Framework for C/C++

The C/C++ Connector Framework is much simpler than the Java one. For example, it does not have its own controller because it uses the Java-Connector Framework controller. Nevertheless it provides a lot of helpful services that the connector plugin can use. The following figure illustrates the framework components.

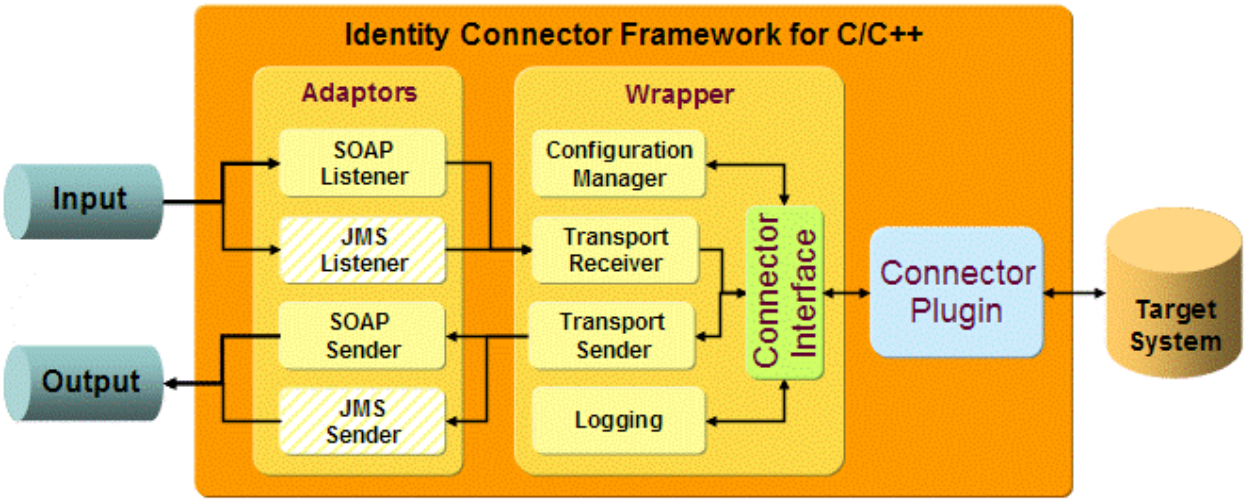



*Figure 26. Identity Connector Framework (C/C++) Components*

Its most important components are:

- **Adaptors** - Listeners and Senders allow communication with either JMS-based or HTTP/SOAP based components.
- A **Configuration Manager** transfers the configuration information from the connectivity configuration to all components, including the connector plugin.
- **Transport Receivers** and **Transport Senders** are responsible for marshalling and un-marshalling the SPML data from streams to C++ SPML objects and vice-versa.
- For each connector plugin, there is a **Wrapper** that loads and instantiates the plugin, supplies the configuration data and relieves the plugin from internal communication handling.

- A **Logging** component provides standard logging mechanisms
- **Connector Interface** - the only interface the connector plugin must recognize. Provides all information in SPML format.
- **Connector Plugin** - the piece of custom code that maps the SPML internal format to the API calls for a specific target system.

# 7. Managing the Connectivity System

This chapter describes how to manage:

- Administrative accounts
- Connectivity security
- Secure connections with SSL
- File handling mechanisms

## 7.1. Managing Administrative Accounts

DirX Identity uses a set of accounts for normal operation.The following figure provides an overview of the most important accounts:

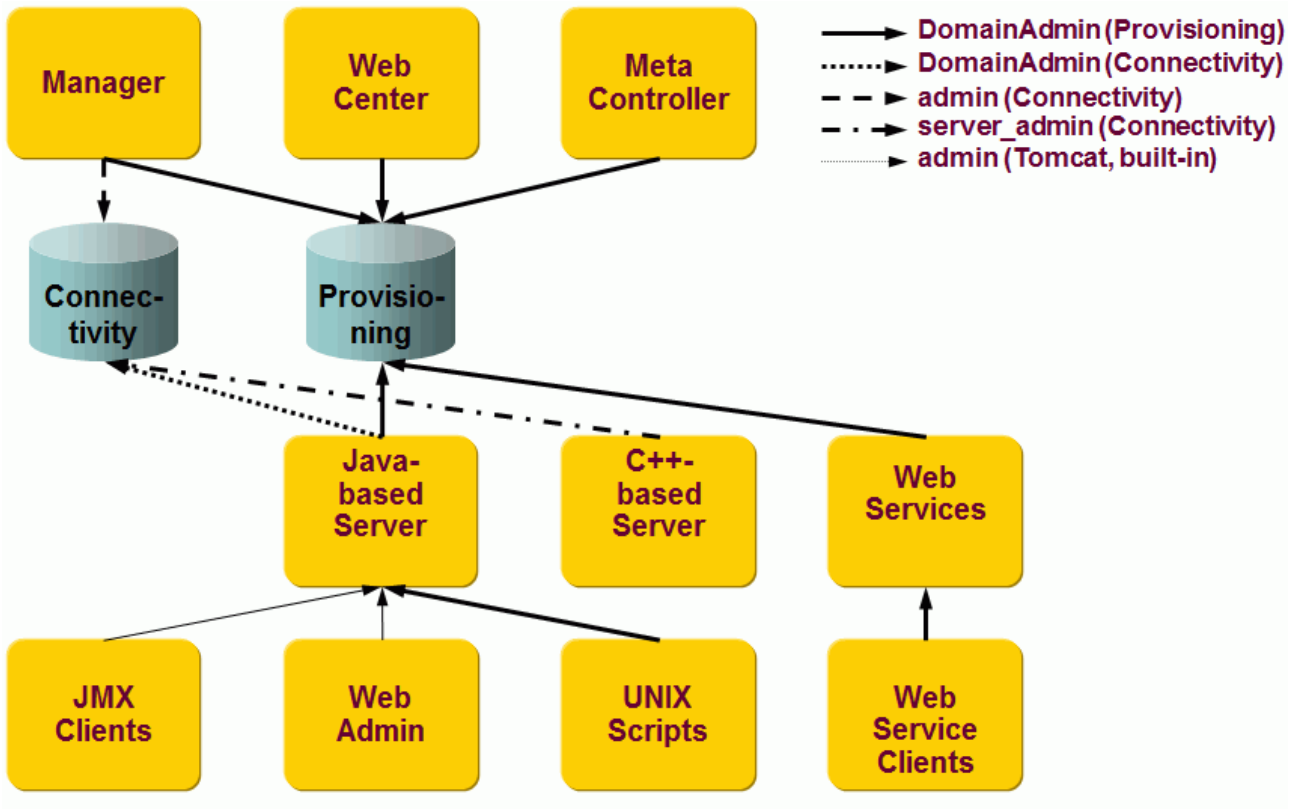



*Figure 27. Connectivity Administrative Accounts*

DirX Identity provides the following administrative accounts:

- **ANYONE (Provisioning, not shown in the figure)** - an account used only for self registration because the user does not yet have an account.This account is used only by the Web Center and has very restricted access rights.
  Its distinguished name is: **cn=ANYONE,cn=*domain***
  **If you change its password in the database, don't forget to change it in Web Center's *password.properties** file, too.
- **DomainAdmin (Provisioning)** - the system account for a specific domain in your Provisioning configuration. It is used by many components like servers, services, agents,

connectors and user interfaces.
Its distinguished name is: *cn=DomainAdmin,cn=**domain*
The initial password is: (the value entered during Initial Configuration)

- **DomainAdmin (Connectivity)** - the corresponding system account for a specific domain in your Connectivity configuration. This account must use the same password as the DomainAdmin account of the Provisioning configuration.
  Its distinguished name is: **cn=DomainAdmin,cn=*** ***domain*****,*dxmC=Users,dxmC=DirXmetahub**
  The initial password is: (the value entered during Initial Configuration for the DomainAdmin (Provisioning))
- **NoApprovalAdmin (Provisioning)** - a special account that can be used by default for assignment of privileges without approval even if the privileges are flagged for approval. This account allows for defining privileges that must be approved manually. Services - especially the Policy Execution service - can use this account to assign privileges without approval.
  Its distinguished name is: *cn=NoApprovalAdmin,cn=**domain*
  The initial password is: (the value entered during Initial Configuration for the DomainAdmin (Provisioning))
- **SystemAdmin (Provisioning)** - the system account for the system domain in your Provisioning Configuration. It is only used by the DirX Identity Manager to copy initial target system definitions from the System Domain to the Customer Domain.
  Its distinguished name is: **cn=SystemAdmin,cn=DirXmetaRole-SystemDomain**
  The initial password is: (the value entered during Initial Configuration)
- **admin (Connectivity)** - an account that has full control over all parts of the Connectivity configuration. Used mainly by the DirX Identity Manager.
  Its distinguished name is: **cn=admin,dxmC=DirXmetahub**
  The initial password is: (the value entered during Initial Configuration)
- **server_admin (Connectivity) -** a restricted account used only by the C-based server. It can read all parts of the configuration database but can only write C-based Server, job and status entry objects.
  Its distinguished name is: **cn=server_admin,dxmC=DirXmetahub**
  The initial password is: (the value entered during Initial Configuration)
- **admin (embedded Tomcat)** - the standard pre-configured Tomcat administrative account to access the Java-based Server's web interface. Note: this embedded Tomcat is not the Tomcat you use to run the Web Center or the Web Services.
- **admin (ActiveMQ message broker)** - the standard pre-configured ActiveMQ administrative account to access the ActiveMQ console.

You should change the passwords for these accounts immediately after you install the C++-based Server using DirX Identity Manager's Data View. Alternatively, you can delete the accounts that you do not need (for example, the WriteAdmin and the ReadAdmin accounts). The next sections provide more information on how to change these passwords.

### 7.1.1. Changing the DomainAdmin Password (Provisioning)

Use any LDAP tool to change the password of this entry: *cn=DomainAdmin,cn=**domain*

Set the identical password value for this entry: **cn=DomainAdmin,cn=*** ***domain*****,*dxmC=Users,dxmC=DirXmetahub**

Set the identical password value for this entry: *cn=NoAppropalAdmin,cn=**domain*

Now change or use the new password at these locations:

- Inform all users of the DirX Identity Manager that use the DomainAdmin account that the password has changed.
- Change the password for the Provisioning Services in the file *install_path**\provisioningServices\spmlv2\conf.xml*
  Note: you can find the related admin DN in the file *install_path**\provisioningServices\spmlv2\conf.xml*
- Change the password for the Web Services in the relevant password file used by the Web Services instance. By default, this is *install_path*/**provisioningWebServices/*** ***instance***/WEB-INF/password.properties* where *instance* is of the form **provisioningServlet-*** ***technical-domain-name*** **(for deployment into the Tomcat server) or *provisioningServlet-*** ***embedded-technical-domain-name*** **(for deployment into the Ids-J Server). However, if you have configured this instance to use a different password file (with the *passwordFile** parameter in *install_path**/provisioningWebServices/***instance***/WEB-INF/web.xml*) then you need to change that password file. Restart the IdS-J server or Tomcat, depending on where you have deployed the Web Services. Note: you can find the related admin DN in the file *install_path**/provisioningWebServices/***instance***/WEB-INF/config.xml*.
- Change the password of the Java-based server in the file *install_path**\ids-j-***domain***-S**n**\private\password.properties*
  Restart the Java-based Server.
  Note: you can find the related admin DN in the file *install_path**\ids-j-***domain***-S**n**\bindprofiles\private\domain.xml*
- Change the password for the Web Center in the file *install_path**\web\webCenter\WEB-INF\password.properties*
  Restart Tomcat or the Web Center servlet.
  Note: you can find the related admin DN in the file *install_path**\web\webCenter\WEB-INF\web.xml*
- Change the password for the DirX Identity REST Service in the file *install_path**/restServices/DirXIdentityRestServices/DirXIdentityRestService-***technical-domain-name***/WEB-INF/password.properties*.
  Restart Tomcat.
  Note: you can find the related admin DN in the file *install_path**/restServices/DirXIdentityRestServices/DirXIdentityRestService-***technical-domain-name***/WEB-INF/web.xml*.
- If the LinkChecker or any scripts use the DomainAdmin as their account, change the password accordingly.

### 7.1.2. Changing the SystemAdmin Password (Provisioning)

It is not necessary to change this user's password because it is only used during

configuration.

### 7.1.3. Changing the Connectivity server_admin Password

Use any LDAP tool to change the password of this entry: **cn=server_admin,dxmC=DirXmetahub**.

Now change or use the new password at these locations:

- Change the password of the C++-based server in the file *install_path***server\conf\dxmmsssvr.ini**
  Restart the C++-based Server.

  you can find the related admin DN in the same file.

- If any scripts use **server_admin** as their account, change the password accordingly.

We recommend the following procedure to change the C++-based server's **server_admin** password:

- Disable all schedules (see the DirX Identity Manager online help for information on how to do this; for example, in the Expert View, right-click the (Central) Configuration object and then click the Scheduler tab. Now click **Help**).
- Check that no workflows are currently running by using the Get Server State function on all servers in your DirX Identity Connectivity domain (see the DirX Identity Manager online help for information on how to do this).
- Stop the DirX Identity IdS-C service (go to the local machines to stop the services).
- Change the password in the directory (use the DirX Identity Manager's Data View or the native tools for each of the directories.
- Change the password entries in the C++-based server initialization file (**dxmmsssvr.ini**).
- Restart the DirX Identity IdS-C service.
- Re-enable the schedules (see the DirX Identity Manager online help for information on how to do this).

### 7.1.4. Changing the Connectivity admin Password

Use any LDAP tool to change the password of this entry: **cn=admin,dxmC=DirXmetahub**.

Now change or use the new password at these locations:

- Inform all users of DirX Identity Manager that use the **admin** account that the password has changed.
- If the LinkChecker or any scripts use **admin** as their account, change the password accordingly.

### 7.1.5. Changing the Embedded Tomcat admin Password

To change the password of the **admin** account of the embedded Tomcat:

- Open a shell window and navigate to *install_path*/ids-j-***domain***-S*n*/bin*.
- Run the script **dxidigest.bat** or **dxidigest.sh** with the new password as an argument. The script will output the new password and its SHA-512 hash value. Password and hash are separated by a colon. For example:

```
abc-
123:0e2f0e2699cacc7f46614a563c09c6d3be376e2e27117facade74810966ad011$
1$9e3ee
7ba0899a66c1bde6adf89879365b249a42b7f772cb4b65e9d6fc848b231cd6eb0d4ef
7762feeacec
1f239e69d25d6b9e908a6b0f8478a9f9393cb07ac9a
```

- Check if the new password was correctly recognized by the script. Some special characters might need escaping when passing the password as argument to the script.
- Navigate to *install_path*/ids-j-***domain***-S*n*/tomcat/conf*.
- Open the file **tomcat-users.xml**.
- Copy the hash from the script output and set it as the password for the **admin** account.

```
<user name="admin"
password="0e2f0e2699cacc7f46614a563c09c6d3be376e2e27117facade74810
966ad011$1$9e3ee
7ba0899a66c1bde6adf89879365b249a42b7f772cb4b65e9d6fc848b231cd6eb0d
4ef7762feeacec
1f239e69d25d6b9e908a6b0f8478a9f9393cb07ac9a"
roles="admin,manager"/>
```

- To ensure that only authorized administrators can modify this file, specify the correct access rights of the file.
- Now change or use the new password at these locations.
- Inform all users who use the Web Admin or any other JMX tool that uses the **admin** account that the password has changed.

### 7.1.6. Changing the ActiveMQ admin Password

To change the password of the **admin** account of the ActiveMQ message broker:

- Open a shell window and navigate to *install_path*/messagebroker/bin*.
- Run the script **dximqdigest.bat** or **dximqdigest.sh** with the new password as an

argument. The script will output the new password, the obfuscated password and its MD5 hash value. For example:

```
abc-123
OBF:1igd1igf1igh16yn1idp1idr1idt
MD5:6351623c8cef86fefabfa7da046fc619
```

- Check if the new password was correctly recognized by the script. Some special characters might need escaping when passing the password as an argument to the script.
- Navigate to *install_path*/**messagebroker/conf**.
- Open the file **jetty-realm.properties**.
- Copy the hash (including the MD5: prefix) from the script output and set it as password for the admin account, for example:

```
admin: MD5:3ad1e9e943bc3431553c0791aac83466, admin
```

- To ensure that only authorized administrators can modify this file, specify the correct access rights of the file.
- Now inform all users who use the ActiveMQ Console that the password has changed.

# 7.2. Managing Connectivity Security

This section discusses general security considerations and describes how to:

- Secure different types of DirX Identity data
- Manage data encryption
- Set up audit signature
- Generate a Personal Security Environment (PSE)
- Manage anonymously readable attributes
- Manage keys

## 7.2.1. Securing DirX Identity Data

By design, DirX Identity assumes that the machines themselves and all of the running processes are protected by operating system-specific security features. The main security weakness is data transfer over the network. The sections in this topic discuss the security requirements for a number of different data types.

#### 7.2.1.1. Securing Control Data

Control data consists of the messages sent by the C++-based Server components via the

messaging service to start, check, and stop processes. Because the server can only start completely configured workflows, there is no opportunity to change any of the configuration data (for example an account, a password, or any other configuration parameter). Consequently, securing this channel is not necessary.

#### 7.2.1.2. Securing Password Changes

Password change data is critical data that should be protected throughout the entire system. Password data is encrypted and only transferred in encrypted form up to the target system interfaces where it is required to be available in clear text form.

Use client-side SSL connections for the messaging service. See the *DirX Identity Installation Guide* for the setup procedure.

Secure the channels to your target systems that transfer password data in clear text form. See "Securing Data to be Synchronized" for more information.

#### 7.2.1.3. Securing Configuration Data

Configuration data is the data that is transferred via LDAP while editing with the DirX Identity Manager. Part of this data is also transferred via LDAP before a workflow or an activity is started. Configuration data is very important and must therefore be handled securely. Connect the C++-based Server via SSL with the configuration database to guarantee secure editing and secure data transfer before a workflow or an activity starts.

#### 7.2.1.4. Securing Administrative Passwords

DirX Identity agents need bind profiles (user name and password) to authenticate to a specific target connected directory. These passwords are very important because they allow you to change a lot of administrative data at the target system side. As a result, administrative passwords must be highly secure. Use the DirX Identity data encryption mechanism to secure these bind profile passwords. See the topic "Managing Data Encryption" for more information.

#### 7.2.1.5. Securing Data to be Synchronized

The category "data to be synchronized" includes data itself and the interfaces used to transfer the data. Use the DirX Identity data encryption mechanism to secure very important data, such as selected attributes. See the topic "Understanding Data Encryption" for more information.

Data interfaces to be secured include:

- **LDAP connections**. Use SSL to secure LDAP connections. See the chapter "Establishing Secure Connections with SSL" for more information.
- **JMS messaging connections**. Use SSL to secure this type of connection. See the chapter "Establishing Secure Connections with SSL" for more information.
- **HTTP connections**. Use SSL to secure HTTP connections. See the chapter "Establishing Secure Connections with SSL" for more information.
- **Files**. Files are often used to transfer data between agents. DirX Identity provides two

mechanisms to transfer files between different machines:

- The file transfer service, which copies data by default via the message service channels. You can secure this connection if you switch on encryption. DirX Identity Manager also uses the file transfer service to allow you to view files from the status areas of other machines. Note that you can use the **Copy to Status Area** flag in the file item object to determine whether critical files are stored in the status area. You can also force DirX Identity to delete these files from the work area if possible (see the **Save Mode** flag in the file item object).
- Shared file systems, for fast file transfer. This connection can only be secured with operating system features.
- **Target system APIs**. The security of these interfaces used by agents is highly dependent on the capabilities of the interface and whether these features are usable through the agent. For more information, refer to the target system documentation or the *DirX Identity Connectivity Reference*.

#### 7.2.1.6. Securing Auditing Information

You can secure auditing information optionally via digital signature. Note that this slows down server operation noticeably. For more information, see the section "Setting Up Audit Signature."

#### 7.2.1.7. About the Crypto Algorithms used by DirX Identity

DirX Identity uses the following types of crypto algorithms:

- Asymmetric RSA up to 2048 bit
- Symmetric TripleDES (56 bit) and RC4 up to 128 bit

### 7.2.2. Managing Data Encryption

This section describes the mechanisms that DirX Identity provides for enabling encrypted data synchronization. DirX Identity allows you to secure the DirX Identity environment and to establish data security, especially password synchronization. DirX Identity can also protect a specific set of attributes via attribute encryption. See the section "Securing DirX Identity Data" for more general considerations.

#### 7.2.2.1. How DirX Identity Data Encryption Works

Directories today permit the storage of passwords in the userPassword attribute either in clear text or one-way encrypted. If you want a secure system, clear text storage is not an option. Choosing one-way encryption no longer permits reading the password. One-way encrypted passwords can only be used for applications that use the directory for authentication. The application delivers the clear text password over an SSL-secured connection to the directory. The directory server encrypts it again and checks it against the stored userPassword value. It then notifies the application whether the comparison was successful, which means that the password is correct.

All target systems force DirX Identity to provide clear text passwords to authenticate. Passwords for accounts must also be set in clear text format. The target system stores the

values in encrypted format. As a result, DirX Identity cannot use the userPassword attribute for password encryption. To enable this kind of functionality, an additional attribute for each user or administrator entry is necessary - dxmPassword - that holds the password in two-way encrypted format.

This method requires that all clients that handle passwords be able to set or update both userPassword and dxmPassword attributes synchronously. To perform this task correctly, the user must authenticate against the directory with the existing password (based on the one-way encrypted userPassword). When the user enters the new password, the client reads the public key from the directory and encrypts the new password value delivered from the user. Now it can set both values in the directory: a replace operation sets the new one-way encrypted userPassword and another replace operation sets the dxmPassword in encrypted format. We assume that all these tasks are performed via secure SSL connections. Otherwise, clear text passwords would be visible in the network. The following figure illustrates DirX Identity secure password synchronization.

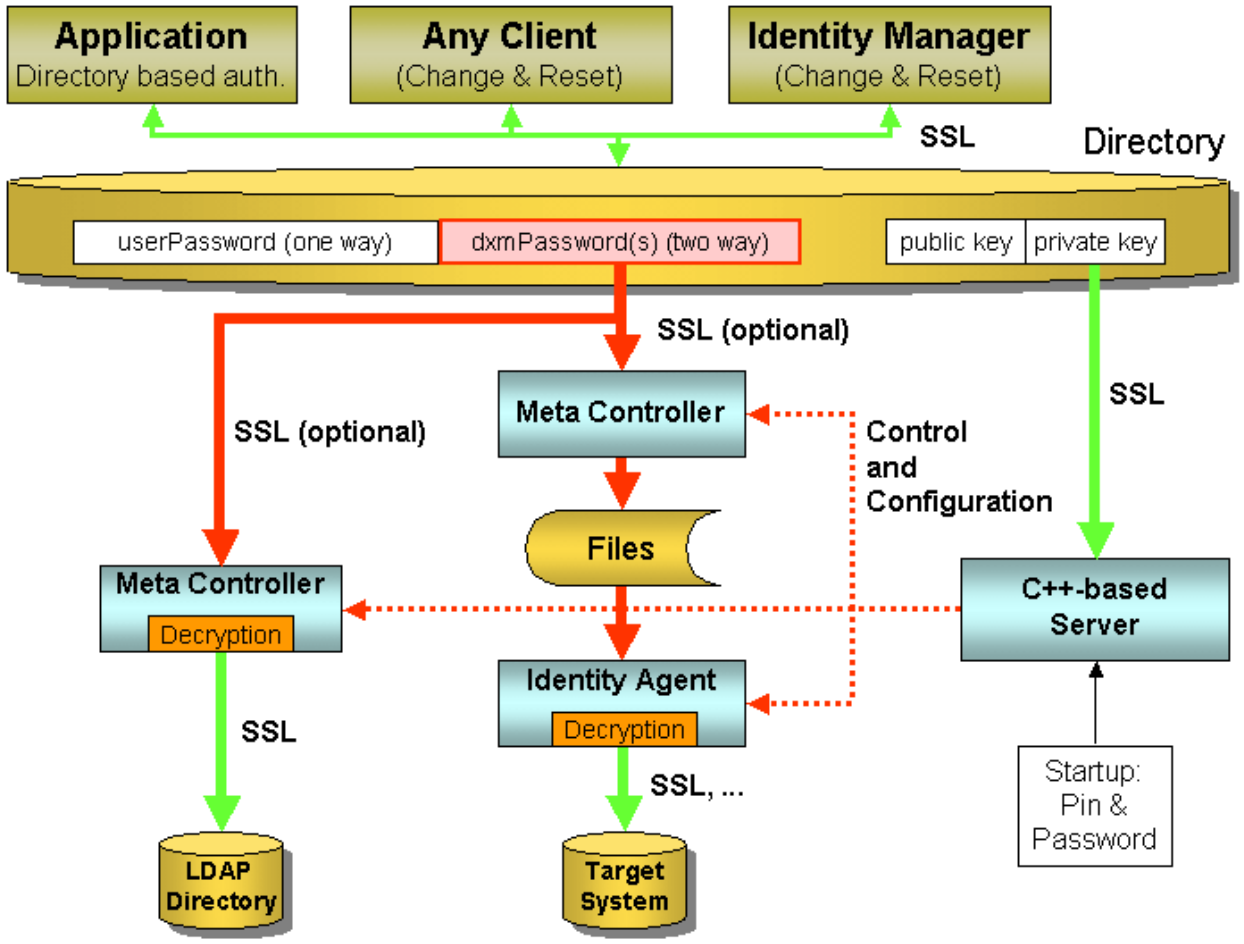



*Figure 28. DirX Identity Secure Password Synchronization*

In the figure, three clients are shown:

- **Any Client** - any client that is able to handle the password synchronization procedure just described can be used to set and maintain password attributes. These clients read the public key, and can be part of a self-service scenario, where users change their passwords regularly or reset their passwords based on additional questions when they have forgotten their password. Alternatively, the password reset can be performed by

an administrator that uses a similar client. Note: These clients are not part of the DirX Identity tool suite.

- **DirX Identity Manager** - DirX Identity provides a password dialog for bind profiles that uses the public key to set and maintain the administrative passwords. Because administrative passwords must be set consistently in the target system and in the corresponding bind profile, there is no built-in protection against changing passwords. If a password is changed on only one side, the corresponding workflow will no longer run. Thus there is no security risk.
- **Application** - any application that uses the directory for authentication. This application only provides the clear text password to authenticate against the directory. Reading the public key is not necessary.

If the dxmPassword fields in the directory are correctly set and encrypted, the C++-based Server can use them. The server must be set up correctly for encryption and the correct **server_admin** password and the PIN for decryption must be entered correctly during startup. The C++-based Server can read the private key from the directory. When starting workflows, several things happen:

- It **processes the configuration files** that still contain encrypted passwords as part of the bind information.
- It **starts the agent** and moves all information necessary for decryption over a secure connection to it.
- The agent is now able to **decrypt the bind password** and to authenticate against the source and target connected directories.
- Additionally the agent can **decrypt any encrypted attributes** (mainly dxmPassword but also any other attributes) if this is the agent that handles the target connected directory API. It can store the decrypted values into the target system.

As you can see in the figure, encrypted information is transferred from the directory in encrypted format up to the last possible point: the target API. Because information is again readable in clear text format here, a **secure connection like SSL** must be used. See the *DirX Identity Connectivity Reference* or the *DirX Identity Meta Controller Reference* for details on how to set up SSL connections.

There are two basic uses of the encryption feature:

- Encryption of administrative passwords only (the bind passwords)
- Encryption of administrative passwords and user passwords or other attributes.



It is not possible to encrypt attributes but not encrypt the administrative passwords. From a security standpoint, this does not make any sense.

#### 7.2.2.2. Handling Specific Encryption Requirements

DirX Identity provides additional features to handle specific encryption requirements:

- You can **disable the encryption feature** for agents that are unable to perform decryption (especially agents integrated into DirX Identity with the Agent Integration

Kit). Use the **Disable Encryption** switch in the corresponding bind profile. When disabling encryption for an agent, you should ensure that no attribute in the attribute configuration object for the agent is marked in the **E** column (encryption column).

- You can use DirX Identity routines for **key management**, if security has been compromised or you want to exchange your key pair and certificate.
- You can set up all file transfers in encrypted mode. In DirX Identity Manager, click Configuration → Messaging Services → Message Service, then set **Encryption Mode** to **Encrypted**.
- You can use the data encryption feature to set up a **password synchronization** scenario. This task requires additional components that are not part of the DirX Identity delivery as previously mentioned, such as clients for user self-service, eventually clients for administrator password reset, and workflows or other mechanisms that detect "old" passwords and notify the user, for example, by e-mail to update the password in the directory. In this scenario, users should not change their passwords directly in the target system. Therefore it makes sense to disable most of the password policies in the target systems and to use common policies in the directory instead. It also makes sense to build extra workflows that run very frequently to update changed passwords in the target systems or to use the different changelog features of the various directories to trigger password update workflows.

You cannot **synchronize the account that is used as administrative bind** (the bind profile for the target system). In future versions of DirX Identity, the agents will generate one-time passwords that are written to the target system and the bind profile in the configuration database in parallel. This method raises the security level.

#### 7.2.2.3. Setting up Data Encryption

Perform the following steps to set up encryption for administrative passwords or data attributes. Steps that are only necessary for the encryption of data attributes are indicated by the notation "data encryption only".

1. **Set up a secure directory server**. Set up your directory server so that it handles the userPassword attribute securely (via one-way encryption). Define the required password policies. Refer to your directory vendor’s documentation for more information about these tasks. Note that the security mechanisms for the directory server are completely independent from DirX Identity. If you do not follow this procedure, passwords are stored in clear text in the directory and state-of-the-art security policies are not used.
2. **Generate a personal security environment (PSE)**. Generate the RSA key pair (public/private) and user certificate and store them in the **cn=server_admin,dxmC=DirXmetahub** entry in the configuration database.
3. **Disable scheduling and wait for workflows that have not completed**. Use the Disable Scheduling and Get Server State functions in the DirX Identity Manager to perform these tasks.
4. **Extend the user data schema** (data encryption only). If user data attributes are to be encrypted, you must extend the user database schema to include the additional attributes (for example, dxmPassword and optionally dxmOldPassword when user passwords must be handled).

5. **Extend or create workflows for encrypted synchronization** (data encryption only). Extend existing workflows or create new workflows to synchronize the encrypted attributes.
6. **Enable encrypted synchronization**. Click the (Central) Configuration object in the DirX Identity Manager Expert View, then click the Server tab. Select either **Administrative Passwords Only** or **Attributes and Administrative Passwords** (data encryption only).
7. **Initially encrypt administrative passwords**. All administrative passwords must be initially encrypted. To perform this task, you can use the change password dialog for the bind profiles in the DirX Identity Manager or run the meta controller's **metacp encryptdata** operation. The **metacp encryptdata** operation takes all current dxmPassword values, encrypts them and stores them in the dxmPassword and dxmOldPassword attributes. The command also sets the userPassword attribute.
8. **Initially encrypt data attributes**. All data attributes (for example, the dxmPassword attribute or other attributes you have selected for encryption) must be encrypted if any values are already in the directory. To perform this task, run the **metacp encryptdata** operation. The operation takes the defined attributes, encrypts them and writes them back into the directory. If you want to set up user passwords and the userPassword values are already one-way encrypted, only a default value can be stored in the dxmPassword attribute. This default is then later synchronized into the target systems and forces the user to change his password (**Note**: this mechanism does not work for target systems where the synchronization needs the old password to change the password, for example RACF).
9. **Configure all C++-based Servers**. Set the **encryptionMode** switch in the **dxmmsssvr.ini** file to **1** to run the server in a secure mode. Set the pin in the *install_path*/ssl/password.properties* file. The server will encrypt the pin in the file during the next start..
10. **Configure SSL for all DirX Identity Managers**.
11. **Configure SSL for all Java-based Servers**. Like the C++-based Server, the pin is read from the *install_path*/ssl/password.properties* file.
12. **Start the C++-based and Java-based Servers**.
13. **Test your workflows**. Start all workflows by hand and check that everything works correctly.
14. **Enable scheduling** - Normal system operation.

The next sections provide more details about these steps.

##### 7.2.2.3.1. Extending/Creating Workflows for Encrypted Synchronization

To configure a synchronization (workflow) for encryption:

- **Define the mapping** for the attribute(s) as usual. For example, suppose you want to transfer user passwords to the Windows NT connected directory. Set up a mapping that maps the dxmPassword to the dxmNTuserPassword.
- **Flag all attributes to be decrypted** in the **E** column of the Attribute Configuration Editor. Note: this is only necessary for the attribute configuration that defines the target API. For example, flag the dxmNTuserPassword attribute in the attribute configuration

of the Windows NT connected directory.

- **Ensure that the connection to your target system is secure**. For example, use SSL for LDAP connections and Kerberos for the connection to the Active Directory.

#### 7.2.2.3.2. Initially Encrypting Administrative Passwords

You can initially encrypt administrative passwords with DirX Identity Manager or with the encryption tool available on the DirX Identity DVD.

You can use the DirX Identity Manager to encrypt each password individually. To encrypt the **admin** password in the Identity Store, perform these steps:

- Click **Global View**.
- Right-click the **Identity Store** connected directory icon and then select **Configure** from the context menu.
- Click the **Bind Profiles** step in the wizard.
- Select **UserAdmin** and open it.
- In **Password**, click .
- Enter the old and new passwords and then click **OK**.

Alternatively, you can use the encryption tool on the DirX Identity DVD. Note that this tool is not installed automatically. To use the tool:

- Save the configuration database (stop all running workflows and disable all schedules).
- Change to the directory **DirXIdentity** → **EncryptionTool** on the DirX Identity DVD.
- Copy the complete directory to your computer.
- Check the settings (**server_address** and **user_pwd (of user "cn=admin,dxmC=DirXmetahub"**)) in the Tcl file **encrypt.tcl**.
- Set the variable **bind_profiles** to TRUE to encrypt **bind_profiles**.
- Run **metacp encrypt.tcl** from the directory you to which you copied the files.
- Check the result in the trace file **tracefile.txt**.

The encryption tool uses the public key in the directory and encrypts the requested fields that are either in clear text or scrambled format.



You can run the encryption tool several times.

#### 7.2.2.3.3. Initially Encrypting Attributes

Use the encryption tool on the DirX Identity DVD to initially encrypt data attributes. Note that this tool is only available on the DirX Identity DVD and is not installed automatically. To use the tool:

- Save the configuration database (stop all running workflows and disable all schedules).
- Change to the directory **EncryptionTool** on the DirX Identity DVD.

- Copy the complete directory to your computer.
- Check the settings (**Server_address**, **User_name**, **User_pwd**, **base_obj**, **subset**, **filter**, and **ldap_attr_list**) in the Tcl file **encrypt.tcl**. (Please note the exact spelling of Server_address, User_name and User_pwd (initial capitalized) in the section for encryption of user data.)
- Set the variable **bind_profiles** to FALSE to encrypt **user data**.
- Run **metacp encrypt.tcl** from the directory to which you copied the files.
- Check the result in the trace file **tracefile.txt**

The encryption tool uses the public key in the directory and encrypts the requested fields that are either in clear text or scrambled format.



You can run the encryption tool several times.

### 7.2.3. Setting up Audit Signature

Perform the following steps to set up digital signature for auditing. Note that you can run auditing without signature (this speeds up server operation).

1. **Generate a personal security environment (PSE)**. Generate the RSA key pair (public/private) and user certificate and store them in the **cn=DomainAdmin,cn=***mydomain***,dxmC=Users,dxmC=DirXmetahub** entry in the configuration database.
2. Set the auditing switches at the password and provisioning **real-time workflow activities** that shall perform auditing in the connectivity configuration database.
3. Set the auditing switches for the **service layer** and r**equest workflows** if you want auditing for these services at the Domain Configuration object in the Provisioning configuration database (see the *DirX Identity Provisioning Administration Guide* or the online help for more information).
4. **Set the signaturePin value** in the file *install_path***\ids-j-***domain*-**S***n***\private\password.properties**.
5. **(Re-)start the Java-based Servers**.
6. **Test auditing**. Check that auditing works correctly for all audit-enabled services.



- You can set up or remove digital signature for auditing at any time. Simply add or delete the certificates at the corresponding **cn=DomainAdmin,cn=*mydomain***,dxmC=Users,dxmC=DirXmetahub* entry in the configuration database.
- You can set up signature only for all auditing services or none. Handling only specific services for signature is not possible.

### 7.2.4. Generating a Personal Security Environment (PSE)

You use the **dirxgenpse** tool to generate a personal security environment (PSE). On Windows systems, you run this tool from the command prompt window.



The **dirxgenpse** tool is only available on the DirX Identity DVD. It is not installed automatically. Copy it from the DVD to the directory *install_path*/bin*.



Do not use the **dirxgenpse** version from the DirX Directory server delivery! It does not work.

dirxgenpse [-*options*]

*Options* are:

**-D** *DN* - specifies the distinguished name (DN) of the keyOwner directory entry (mandatory)

**-P** *PSE-PIN* - specifies the PSE protection passphrase (mandatory)

**-s** *serial* - specifies the serial number of the certificate (mandatory for the first key pair; if omitted, it is set to the next higher value)

**-l** *keylen* - specifies the length of the RSA key in bits (default is 1024)

**-O** - runs the tool in off-line mode (the default is online)

**-h** *host* - specifies the host name of the LDAP server (default is localhost)

**-p** *port* - specifies the LDAP port of the LDAP server (default is 389)

**-w** *pwd* - specifies the password of the keyOwner directory entry (optional)

**-t** *secpath* - specifies the path to the **cert8.db** file for SSL/TLS security (default is none)

**-e** *expiry* - specifies the expiration period in days (default is 1826)

**-o** *file* - specifies the filename of the PSE files to be created (optional)

**-v** - runs the tool in verbose mode (default is no verbose)



You must supply the distinguished name (DN) of the keyOwner and a PSE Protection PIN.

There is a conflict between an internally-used private key and using the key version number **00000001** for a customer's private key. This conflict is relevant for the data encryption case. Therefore, if you start using encryption, you must start with a serial number 2 (00000002) by using the **-s** *serial* option. You don't need to use this option for subsequent generate operations because the next free number is chosen automatically.

If you already have at least one private key with a serial number **00000001** in the PSE attribute you must resolve the conflict by migrating to a new private key; you cannot have any private key with serial number **00000001**. Even if you have a current private key with serial number **00000002** but a previous private key with serial number 00000001, you must generate a new private key.

If you generate a PSE with the same serial number as an existing PSE, the outcome is undefined and the Identity services will report errors.



Use secure PINs to ensure high security and do not forget the values! There is absolutely no way to recover them.

To run the **dirxgenpse** tool with the following examples, open a command prompt window in the DirX Identity installation bin subfolder *install_path*\bin*.

Example for data encryption:

```
dirxgenpse -D cn=server_admin,dxmc=DirXmetahub -s 2 -P
mYverYspeciaLpiN$41
```

This command generates a key pair and a certificate based on the PIN value **mYverYspeciaLpiN$41** with serial number 2 and writes it into the **server_admin** entry.

Example for audit signature:

```
dirxgenpse -D
cn=DomainAdmin,cn=MyDomain,dxmC=Users,dxmC=DirXmetahub -P
mySignaturePin$7
```

This command generates a key pair and a certificate based on the PIN value **mySignaturePin$7** and writes it into the **DomainAdmin** entry in the Connectivity configuration.

To find the serial number of the private key, in DirX Identity Manager, open the Connectivity View and go to the Data View. On the Tree tab, select the entry of cn=server_admin,dxmc=DirXmetahub. In the right pane under All Attributes you see then the keyOwnerPSE attribute. The value has a readable prefix like

```
{P:1:3:2:IR9m5usbwteo4g9YgXZwmdP0dzY=:00000003:cn=server-
admin,1.3.12.2.1107.1.3.102.4.13.17=DirXmetahub}
```

The number, here **00000003**, before cn=server-admin is the serial number of the private key.

### 7.2.5. Managing Anonymously Readable Attributes

To make it easier to set up DirX Identity in a distributed environment, DirX Identity provides a set of attributes in the Connectivity configuration database that is readable with anonymous bind. Consequently, you cannot disable anonymous access of your directory servers. The list of attributes is:

**Object class: dxmRoot**

- dxmSpecificAttributes (DXMSPECAT)

**Object class: dxmATS**

- dxmService-DN (DXMSV)
- dxmQueueManager (DXMQMGR)
- dxmStreamCommand (DXMSTCMD)
- dxmStreamFileService (DXMSTFS)
- dxmStreamStatusTracker (DXMSTST)
- dxmStreamEvent (DXMSTEVNT)
- dxmSupportedTopics (DXMSTOPICS)

**Object class: dxmService (DXMSV)**

- dxmAddress (DXMADR)
- dxmDataPort (DXMDPORT)
- dxmSecurePort (DXMSPOPRT)

**Object class: dxmTopic**

- dxmTopicName (DXMTOPICNAME)
- dxmTopicValue (DXMTOPICVAL)

**Object class: dxmCentralConfig**

- dxmEncryptionMode (DXMENM)

**Object class: keyOwner**

- userCerificate (UC)

## 7.2.6. Managing Keys

This section gives hints on how to manage keys within DirX Identity, including:

- Managing keys for data encryption
- Using the key migration tool
- Managing keys for audit signature

### 7.2.6.1. Managing Keys for Data Encryption

You should change your PSE for data encryption from time to time or when security has been corrupted. DirX Identity provides smooth certificate exchange. This is especially important in environments that use the DirX Identity Windows Password Listener. It keeps its own local certificate. Due to network problems the update of this local certificate might be delayed. Thus, the whole environment (including all DirX Identity agents) must be able to work with old and new encrypted data values.

The procedure for key exchange consists of these steps

- Disable all schedules.
- Check that no workflows are running.
- Stop the DirX Identity IdS-C and IdS-J services.
- Create a new PSE (key pair and certificate) with a new pin. This replaces the certificate in the directory which means that all clients that encrypt data values work from this time with the new public key.
- Change the *install_path*/ssl/password.properties* file for all servers. Set the previousPin to the value of the PIN (the PIN of the previous key) and set the PIN to the new PIN value (used to create the new key).
- Start the services. The servers (and for the C++-based server, all agents that are started via workflows) can now handle two private keys for data decryption.
- The Java-based Server's configuration handler will distribute the new certificate (public key) to all Windows Password Listeners. It does not matter how long this will take as long as the old private key is available in the directory.
- Run the key migration tool to convert all encrypted values from the old key to the new key. Run this tool from time to time.
- Before you intend to change the PSE again, run the key migration tool to check that there are no old encrypted values in your directory (this would indicate that there is a Windows Password Listener that did not get the new certificate). Check the reason and repair this problem. When the key migration tool does not find any old encrypted values in the directory, you can delete the old private key in the directory and start the key exchange procedure again.



DirX Identity can only handle two private keys in the directory (this requires the corresponding two PIN values during server startup). Be sure that no data values encrypted with the old key exist in the directory before you change the PSE again. Otherwise the agents will not be able to decrypt these values and the workflows will fail.

#### 7.2.6.2. Using the Key Migration Tool

DirX Identity provides a key migration tool to convert all encrypted values from an old key to a new key. Data that is encrypted with the old key is decrypted using the old private key (the PIN must be provided) and is encrypted with the current public key (from the certificate). Data in clear text is processed as in the initial encryption.

Note: The scripts can be run several times. Actual data is recognized and is not encrypted again.

To use the key migration tool:

- Change to the directory **EncryptionTool** on the DirX Identity DVD.
- Copy the complete directory to your computer.
- Check the settings in the Tcl file **encrypt.tcl**. Set the variable **bind_profiles** to either encrypt **bind_profiles** or **user data**.

- Run the script: **metacp encrypt.tcl** *old_sequence_number* | **PREVIOUS**
- The script prompts you for the PIN of the old key with the given sequence number *old_sequence_number*. If the argument **PREVIOUS** is given the PIN for the second-newest key is needed.
- Check the result in trace file **tracefile.txt**

#### 7.2.6.3. Managing Keys for Audit Signature

You should change your PSE for audit signature from time to time or when security has been compromised.The procedure for key exchange consists of these steps:

- Export the **cn=DomainAdmin,cn=***mydomain***,dxmC=Users,dxmC=DirXmetahub** entry to back up your current certificate for audit signature.If you use tools to verify the signature of auditing information, be sure to keep these certificates; otherwise you will no longer be able to verify the auditing information.
- Stop the DirX Identity IdS-J service(s) (stops the Java-based Servers).
- Create a new PSE (key pair and certificate) with a new PIN.This action replaces the certificate in the directory, which means that all clients that encrypt data values work from this time with the new public key.
- Change the **signaturePin** value in the **password.properties** files for all Java-based Servers.
- Start the DirX Identity IdS-J service(s).(Starts the Java-based Servers.)
- Add the new certificate information to your tools that you use to verify the signature of auditing information.

## 7.3. Establishing Secure Connections with SSL

Secure Socket Layer (SSL) is a common technique for encrypting TCP/IP network communication based on "trusted" certificates or "keys".Transport Layer Security (TLS) is a small evolutionary advance over SSL and is intended to replace SSL at some point.Whether TLS or SSL is used in a particular connection is the result of a negotiation between client and server.Possibly more important, the key length is negotiated, too.A key length of 40 bits is considered "weak", while a key length of "128" bit is considered "strong".

There are two security stages:

Stage 1 ("**certificate-based server authentication**") - if you want to make sure the server you are talking to is actually the one you think it is.Stage 1 implies encryption.It is often referred to as "server-side SSL".

Stage 2 ("**certificate-based client authentication**") - if the server asks you to authenticate yourself through a certificate.Stage 2 implies Stage 1. It is often referred to as "client-side SSL".

For in-depth information on this topic, please refer to https://docs.oracle.com/javase/8/docs/technotes/tools/#security (search for the appropriate key words; for example, "+keytool +keystore +cacerts").

DirX Identity components use a lot of LDAP and HTTP connections either for internal communication or for communication with external target systems. These connections can be secured via SSL for optimum security.

This chapter gives an overview on the available connections and detailed information about how to set them up. The next figure shows DirX Identity's components and the SSL connections that can be made between them.

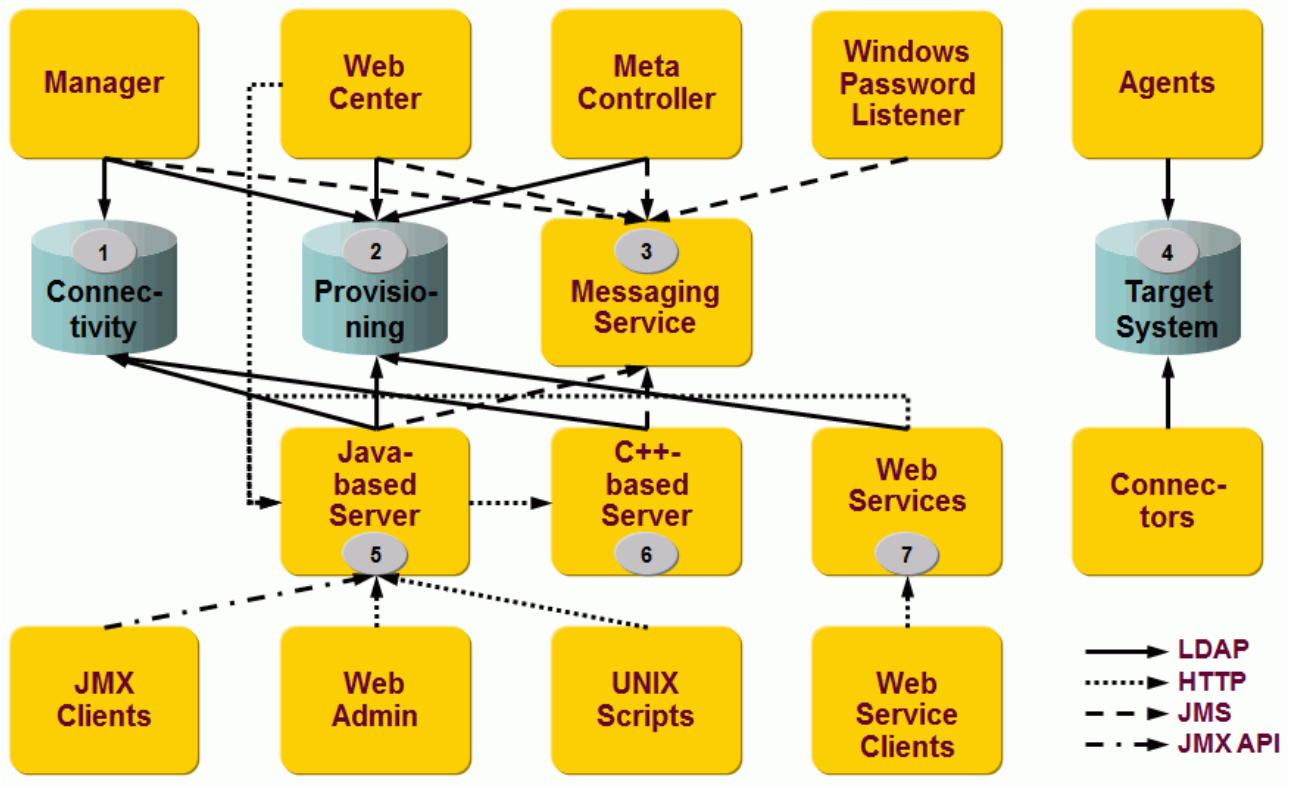



*Figure 29. Overview of SSL Connections*

You can configure SSL for each server component. Note that SSL connections require some extra time during the bind operation. Therefore, we recommend that you set up SSL connections only for distributed components.

The next sections describe the procedures for securing connections between each component shown in the figure, in Windows notation. For UNIX, the steps are slightly different:

- Use the scripts *.sh instead of *.bat
- Use forward slashes (/) instead of backward slashes (\)

### 7.3.1. Securing Connectivity Database Connections with SSL

Running the Connectivity configuration database with SSL requires setting up the directory server (see the corresponding server documentation) and all of the clients that access it, including:

- LDAP server
- DirX Identity Manager

- Java-based Server
- C++-based Server
- Java-based configuration wizard

The next sections explain how to set up these components for SSL.

#### 7.3.1.1. Setting up the LDAP Server

This section explains how to set up the Connectivity configuration LDAP server for SSL.

##### 7.3.1.1.1. Managing Keys

DirX Directory comes with an example server and CA certificate for testing purposes. This keychain is available in the file *dirx_install_path*/LDAP/conf/cert_ldapserver.pem*.

For a production environment, you should set up your own key material for the LDAP server.

##### 7.3.1.1.2. Preparing the LDAP Server for SSL

To establish an SSL connection on the LDAP server side for DirX Directory:

- Import the file *dirx_install_path***/LDAP/conf/cert_ldapserver.pem** into the LDAP SSL configuration using the DirX Directory Manager. Check the DirX Directory documentation to verify how to set up the DirX Directory for SSL.



If you see a file named *dirx_install_path*/LDAP/conf/cert_ldapserver.pem.new*, use this file instead.

##### 7.3.1.1.3. Important Locations

Server certificate chain: *dirx_install_path***/LDAP/conf/cert_ldapserver.pem**.
Password for server certificate: **dirxdirx** (default)

#### 7.3.1.2. Setting up DirX Identity with Initial Configuration

This section explains how to set up DirX Identity with Initial Configuration to use an SSL-secured connection to the Connectivity configuration LDAP server.

##### 7.3.1.2.1. Managing Keys

DirX Identity comes with different files containing the test certification authority (CA) certificate that has signed the test server certificate used from DirX Directory for incoming LDAP connections. These files can be used to establish an example SSL/TLS connection from the DirX Identity components to a DirX Directory LDAP directory. For other directory types, see the vendor documentation.

The following files are provided in the path *install_path***/client/conf**:

- **cert8.db** - contains the DirX Directory test certificate "cn=test-CA,ou=DirX-

Example,o=My-Company". The file **key3.db** must be also present in order to use the file **cert8.db**. These files are used by the C/C++ components of DirX Identity (C++-based Server, Meta Controller) to establish LDAP-secured connections.

The following files are provided in the path *install_path***/GUI/**:

- **cacerts** - contains the DirX Directory test certificate "cn=test-CA,ou=DirX-Example,o=My-Company". This file is the default java certification store used from the DirX Identity Manager.

##### 7.3.1.2.2. Preparing the Certificate Stores

To establish secured SSL connections to the LDAP directory, the CA certificate that has signed the LDAP server certificate must be imported into different certificate stores.

Use DirX Identity Manager to export the DirX Directory test-CA certificate from *install_path***/GUI/cacerts**:

- At the start, you have no configured connection. In this case select Data View and then click one or two times on **Cancel**.
- In the Tools menu, select **Options**.
- Select **This application's installation folder** (the file *install_path***/GUI/cacerts** is shown) and then export the certificate **testca** to a file **testca.cer**.

Use DirX Identity Manager to import the DirX Directory CA certificate to the Java trust stores in use:

- In the Tools menu, choose **Options**.
- Select **Java Runtime Environment**.
- Select **Import**, and then select the LDAP CA certificate you want to import (**testca.cer**). When prompted for the key store password, enter the default value **changeit**.
- Select **Other** and then select the trust store used from your Tomcat installation (either *tomcat_java_home***/lib/security/cacerts** or Tomcat's trust store if it is already using SSL).
- Select **Import** and then select the LDAP server certificate you want to import ( **testca.cer**).
- When prompted for the key store password, enter the default value **changeit**.
- Click **OK** and the certificate will be imported.

Alternatively, you can use the **keytool** command to export and import the CA certificate. For detailed information on the keytool command, please refer to https://docs.oracle.com/javase/8/docs/technotes/tools/#security.

The C++-based Server will use *install_path***/client/conf/cert8.db** as the certificate store. This store already contains the DirX Directory test-CA certificate. If you need to add your own certificate, you need to administer this file with the **certutil** tool, which is part of the Mozilla NSS Utils.

#### 7.3.1.2.3. Using Initial Configuration to Set up DirX Identity

Start the Initial Configuration. In the Connectivity Configuration step, check **Use SSL** and configure the correct LDAP SSL port. This action configures every DirX Identity component to use SSL to the LDAP Connectivity directory.

After running Initial Configuration, use DirX Identity Manager's Connectivity view to set up the Identity Store of your domain to use SSL:

- Select the Global View.
- Right-click the Identity Store of your domain and then select **Configure**.
- In **General Information**, display the properties of the Service object.
- Select **SSL** and the correct **Secure Port** and then click **OK**.
- In **Bind Profile**, display the properties of the Domain_Admin object.
- Select **SSL Connection** and then click **OK**.
- Click **Finish**.

The next sections describe the configuration in detail for each single component.

#### 7.3.1.2.4. Important Locations

Trust store: *install_path***/client/conf/cert8.db.**

Trust store: *install_path***/GUI/cacerts**.
Password for trust store: **dirxdirx** (default).

Trust store: *tomcat_java_home***/lib/security/cacerts.**
Password for trust store: **changeit** (default).

Trust store: *dxi_java_home***/lib/security/cacerts**.
Password for trust store: **changeit** (default).

### 7.3.1.3. Setting up Identity Manager

This section describes how to set up DirX Identity Manager for SSL to the Connectivity configuration database. The DirX Identity Manager requires LDAP / LDAPS access to the Connectivity configuration database to manage it.

#### 7.3.1.3.1. Managing Keys

The DirX Identity Manager uses its own trust store named *install_path***/GUI/cacerts**. By default, this key store already contains the LDAP server certificate provided with the installation.

If you need to use another server certificate, import it using the DirX Identity Manager. In the **Tools** menu, select **Options**. On the next page, the GUI's key store is already selected by default (**This application's installation folder** is already selected and the file *install_path***/GUI/cacerts** is shown.) Choose **Import** and then select the certificate you want to import.

Alternatively, you can use the **keytool** command to import the server certificate. For detailed information on the **keytool** command, please refer to https://docs.oracle.com/javase/8/docs/technotes/tools/#security.

##### 7.3.1.3.2. Setting up Manager for SSL (server-side SSL)

For SSL connections over LDAP to the Connectivity configuration database, you must:

- Make sure that a suitable CA certificate is available in *install_path*/**GUI/cacerts**. "Suitable" means that the certificate transmitted by the directory server must be trusted by at least one of the certificates stored in the trust store cacerts (for details, see the "Managing Keys" section).
- Make sure that the bind profiles you are using have the correct SSL port set and the flag **Use secure connection (LDAP v3 only)** is turned on.

##### 7.3.1.3.3. Important Locations

Trust store: *install_path*/**GUI/cacerts**.
Password for trust store: **dirxdirx** (default).

### 7.3.1.4. Setting up Web Center

This section describes how to set up DirX Identity Web Center for SSL to the Connectivity configuration database. Web Center requires LDAP / LDAPS access to the Connectivity configuration database to read some configuration data. Note that server, port and SSL flag for connections to the Connectivity configuration database are taken from the domain object in the Provisioning database.

#### 7.3.1.4.1. Managing Keys

Store the LDAP server certificate either in the trust store of the Java environment that Tomcat is using or use the Tomcat trust store.

To import the server certificate, use DirX Identity Manager:

- In the Tools menu, select **Options**.
- On the next page, the GUI's trust store is selected by default. Select **This application's installation folder** (the file *install_path*/**GUI/cacerts** is shown) and export the certificate **testca** to a file **testca.cer**.
- Select **Other** and then select the appropriate trust store (either *tomcat_java_home*/**lib/security/cacerts** or Tomcat's trust store if it is already using SSL).
- Select **Import** and then select the LDAP server certificate you want to import: in this case, the previously exported **testca.cer**.
- When prompted for the key store password, enter the default value **changeit**.
- Click **OK** and the certificate will be imported.

Alternatively, you can use the **keytool** command to import the CA certificate. For detailed information on the **keytool** command, please refer to https://docs.oracle.com/javase/8/docs/technotes/tools/#security.

If you decide to create a new trust store for Tomcat, customize your Tomcat configuration (for details, see the Tomcat description).

Windows example:

- Activate the Tomcat Configuration Menu (**Start** → **Programs** → **Apache Tomcat** *version* → **Configure Tomcat**).
- Click the **Java** tab.
- Add **-Djavax.net.ssl.trustStore=***truststore_path* **-Djavax.net.ssl.trustStorePassword=***truststore_password* to the Java **Options**.

UNIX example:

- Add **-Djavax.net.ssl.trustStore=***truststore_path* **-Djavax.net.ssl.trustStorePassword=***truststore_password* to your Tomcat environment variable **JAVA_OPTS**.

##### 7.3.1.4.2. Important Locations

Trust store: *tomcat_java_home***/lib/security/cacerts** or the trust store already used by Tomcat.
Password for trust store: **changeit** (default).

#### 7.3.1.5. Setting up the Java-based Server

This section describes how to set up the Java-based Server for SSL to the Connectivity configuration database. The Java-based Server requires LDAP / LDAPS access to the Connectivity configuration database to read the server configuration, the real-time workflow definitions and other related configuration objects.

##### 7.3.1.5.1. Managing Keys

For each Java-based Server that is to use an SSL connection, store the LDAP server certificate in the key store *dxi_java_home***/lib/security/cacerts** of the Java for DirX Identity.

To import the server certificate, use the DirX Identity Manager:

- In the Tools menu, select **Options**.
- Select **This application's installation folder** (the file *install_path***/GUI/cacerts** is shown) and export the certificate **testca** to a file **testca.cer**.
- Select **Java Runtime Environment**.
- Select **Import** and then select the LDAP CA certificate you want to import (**testca.cer**). When prompted for the key store password, enter the default value **changeit**.
- Click **OK** and the certificate will be imported.

Alternatively, you can use the **keytool** command to import the server certificate. For detailed information on the **keytool** command, please refer to https://docs.oracle.com/javase/8/docs/technotes/tools/#security.

#### 7.3.1.5.2. Setting up the Java-based Server for SSL

For each Java-based Server that is to use an SSL connection:

- Make the following changes to the files *install_path*/**ids-j-***domain*-**S***n*/**bin/bindcredentials.xml** and *install_path*/**ids-j-***domain*-**S***n*/**bindprofiles/private/domain.xml**:
  Switch the ssl flag from false to true: <ssl>true</ssl>.
- Update the port number and change to the ssl port; for example, <port>636</port>.
- Make sure that a suitable server certificate is available in the key store of the Java for DirX Identity. (For details, see the "Managing Keys" section).

#### 7.3.1.5.3. Important Locations

Trust store: *dxi_java_home*/**lib/security/cacerts**.
Password for trust store: **changeit** (default).

### 7.3.1.6. Setting up the C++-based Server

This section describes how to set up the C++-based Server for SSL to the Connectivity configuration database. The C++-based Server requires LDAP / LDAPS access to the Connectivity configuration database to read the server configuration as well as Tcl-based workflow definitions and other related configuration objects.

#### 7.3.1.6.1. Managing Keys

For each C++-based Server that is to use an SSL connection:

- Ensure that you're using the correct **cert8.db** file. By default, the C++-based Server uses the file **cert8.db** located in *install_path*/**client/conf**. If you use your own key material, please provide your version of **cert8.db**. You always need the additional file **key3.db** in order to use **cert8.db**.

#### 7.3.1.6.2. Setting up the C++-based Server for SSL

For each C++-based Server that is to use an SSL connection:

- Open the file dxmmsssvr.ini in the path *install_path*/**server/conf**.
- Set the parameter **port=***secure_port*, for example port=636.
- Set the parameter **ssl=1**.
- Set the parameter **cert-db-path** to the location of the **cert8.db** (typically in the folder *install_path*/**client/conf**).
- Restart the server. It should connect to the DirX LDAP directory via SSL (see the DirX documentation for information on how to check this action).

Note: Do not set the DirX environment variable DIRX_TRUSTED_CA.

#### 7.3.1.6.3. Important Locations

Trust store: *install_path*/**client/conf/cert8.db**.

### 7.3.1.7. Setting up the Java-based Configuration Wizard

This section describes how to set up the Java-based configuration wizard for SSL to the Connectivity configuration database. The Java-based configuration wizard requires LDAP / LDAPS access to the Connectivity configuration database to read and update all kinds of configuration data and to load the Connectivity schema during the Initial Configuration or Update Configuration run.

#### 7.3.1.7.1. Managing Keys

If the configuration wizard is to use an SSL connection, store the LDAP server certificate in the key store *dxi_java_home*/**lib/security/cacerts** of the Java for DirX Identity.

To import the server certificate, use the DirX Identity Manager:

- In the Tools menu, select **Options**.
- Select **This application's installation folder** (the file *install_path*/**GUI/cacerts** is shown) and then export the certificate **testca** to a file **testca.cer**.
- Select **Java Runtime Environment**.
- Select **Import** and then select the LDAP CA certificate you want to import **(testca.cer**). When prompted for the key store password, enter the default value **changeit**.
- Click **OK** and the certificate will be imported.

Alternatively, you can use the **keytool** command to import the server certificate. For detailed information on the **keytool** command, please refer to https://docs.oracle.com/javase/8/docs/technotes/tools/#security.

#### 7.3.1.7.2. Setting up the Java-based Configuration Wizard for SSL

If the configuration wizard is to use an SSL connection:

- Check **Use SSL** in the **Connectivity Directory** step.
- Update the port number in this step to the LDAP Server's SSL port.
- Make sure that a suitable server certificate is available in the key store of the Java for the configuration wizard itself. (For details, see the "Managing Keys section").
- Make sure that a suitable server certificate is available in the **metacp** tool's **cert8.db** trust store (see the section "Important Locations") because **metacp** also uses SSL if the **Use SSL** flag is set for loading LDIF files to the Connectivity database during the configuration wizard run.

#### 7.3.1.7.3. Important Locations

Trust store for the configurator itself: *dxi_java_home*/**lib/security/cacerts**.
Password for trust store: **changeit** (default).

Trust store for **metacp**: *install_path*/**client/conf/cert8.db**.

## 7.3.2. Securing Provisioning Database Connections with SSL

Running the Provisioning configuration database with SSL requires setting up the directory server (see the corresponding server documentation) and setting up all the clients that access this server, including:

- LDAP server
- DirX Identity Manager
- Web Center
- Java-based Server
- Web services
- Meta controller
- Java-based configuration wizard

The next sections explain how to set up these components for SSL.

#### 7.3.2.1. Setting up the LDAP Server

For a description of how to set up the Provisioning configuration LDAP server for SSL, see the section "Setting up the LDAP Server" in the section "Securing Connectivity Database Connections with SSL".

#### 7.3.2.2. Setting up DirX Identity with Initial Configuration

For a description of how to easily set up DirX Identity to use SSL with the Provisioning configuration LDAP server, see the section "Setting up DirX Identity with Initial Configuration" in the section "Securing Connectivity Database Connections with SSL".

#### 7.3.2.3. Setting up Identity Manager

The DirX Identity Manager requires LDAP / LDAPS access to the Provisioning configuration database to manage it.

For SSL connections over LDAP to the Provisioning configuration database, see the section "Setting up Identity Manager" in the section "Securing Connectivity Database Connections with SSL".

#### 7.3.2.4. Setting up Web Center

This section describes how to set up DirX Identity Web Center for SSL to the Provisioning configuration database. The Web Center requires LDAP / LDAPS access to the Provisioning configuration database to manage its objects.

##### 7.3.2.4.1. Managing Keys

See "Setting up Web Center" in the section "Securing Connectivity Database Connections

with SSL".

#### 7.3.2.4.2. Setting up Web Center for SSL (server-side SSL)

To set up an SSL connection:

- Open the file *install_path*\web\webCenter-*domain*\WEB-INF\web.xml.
- Adapt the parameter **com.siemens.webMgr.ldap.port** to the server SSL port (for example, **636**).
- Set SSL mode via the parameter **com.siemens.webMgr.ldap.ssl** to "true".
- Restart Tomcat.

#### 7.3.2.4.3. Setting up Web Center for Password Management for SSL (server-side SSL)

To set up an SSL connection:

- Open the file *install_path*\web\pwdManagement-*domain*\WEB-INF\web.xml.
- Adapt the parameter **com.siemens.webMgr.ldap.port** to the server SSL port (for example, **636**).
- Set SSL mode by setting the parameter **com.siemens.webMgr.ldap.ssl** to "true".
- Restart Tomcat.

### 7.3.2.5. Setting up the Java-based Server

The Java-based Server requires LDAP / LDAPS access to the Provisioning configuration database to read the request workflow definitions and other related configuration objects. For instructions on how to set up SSL connections over LDAP to the Provisioning configuration database, see the section "Setting up the Java-based Server" in the section "Securing Connectivity Database Connections with SSL".

### 7.3.2.6. Setting up the Web Services

This section describes how to set up the Web Services for SSL to the Provisioning configuration database. The Web Services require LDAP / LDAPS access to the Provisioning configuration database to manage their objects.

#### 7.3.2.6.1. Managing Keys

The managing keys task depends on where you have deployed the Web Services:

- If you have deployed the Web Services into the Tomcat of the Java-based Server, you must store the LDAP server certificate in the key store *dxi_java_home*/lib/security/cacerts of the Java for DirX Identity. For details, see the section "Setting up the Java-Based Server" in the section "Securing Connectivity Database Connections with SSL".
- If you have deployed the Web Services into an external Tomcat, see the section "Setting up Web Center".

#### 7.3.2.6.2. Setting up the Web Services for SSL

- Modify the file *install_path***/provisioningWebServices/***instance***/WEB-INF/config.xml**, where *instance* is of the form **provisioningServlet-***technical-domain-name* (for deployment into the Tomcat server) or **provisioningServlet-embedded-***technical-domain-name* (for deployment into the Ids-J server). Modify said file so that **port** is **636** and **ssl** is **true**.
- Restart the IdS-J server or Tomcat, depending on where you have deployed the Web Services.

#### 7.3.2.6.3. Important Locations

If you have deployed the Web Services into the Tomcat of the Java-based Server, see the section "Setting up the Java-Based Server" in the section “Securing Connectivity Database Connections with SSL”.

If you have deployed the Web Services into an external Tomcat, see the section "Setting up Web Center for SSL to the Provisioning Configuration".

### 7.3.2.7. Setting up the Meta Controller

The meta controller (**metacp**) requires LDAP / LDAPS access to the Provisioning configuration database to provision and synchronize objects. For instructions on setting up SSL connections from the meta controller to the Provisioning configuration database, see the *DirX Identity Meta Controller Reference*.

By default, the file **cert8.db** located in *install_path***/client/conf** is used. If you use your own key material, please provide your version of **cert8.db**. You always need an additional file **key3.db** in order to use **cert8.db**.

To use SSL within Tcl-based workflows, you must enable the flag **SSL Connection** in the corresponding bind profiles.

#### 7.3.2.7.1. Important Locations

Trust store: *install_path***/client/conf/cert8.db**

### 7.3.2.8. Setting up the Java-based Configuration Wizard

The Java-based Configuration Wizard requires LDAP / LDAPS access to the Provisioning Configuration database to load the Provisioning schema and the domain data to the Provisioning Configuration database during the Initial Configuration or Update Configuration run.

For instructions on how to set up SSL connections over LDAP to the Provisioning Configuration database, see the section "Setting up the Java-based Configuration Wizard" in the section "Securing Connectivity Database Connections with SSL". Follow the instructions given in the section, but set the **Use SSL** flag in the **Provisioning Directory** step instead of the **Connectivity Directory** step.

### 7.3.3. Securing Identity Server Connections with SSL

Running all Identity servers - the Java-based Server, the C++-based Server and the ActiveMQ Message Broker - with SSL requires setting up the servers themselves and all clients that access the servers, including:

- DirX Identity Manager
- Java-based Identity Server
- C++-based Identity Server
- ActiveMQ Message Broker and Web Console
- Web Admin and Server Admin
- Web Center
- Business User Interface
- Supervisor
- Web Services
- Windows Password Listener
- Meta controller
- JMX clients
- UNIX scripts

Note that DirX Identity mainly uses server-side SSL. The only exception is communication with the ActiveMQ Message Broker, which also requires client-side SSL. The next sections describe how to set up these components for SSL.

Securing the Identity servers consists of two steps:

- Setting up the X.509 certificates
- Securing the Identity services

These steps are described in the following sections.

#### 7.3.3.1. Setting up the X.509 Certificates

An X.509 certificate provides a way of binding an identity (in the form of an X.500 distinguished name) to a public key. The certificate essentially consists of an identity concatenated with a public key, with the entire certificate being digitally signed in order to guarantee the association between the identity and the public key.

The certificate must be signed by someone that you trust. The certificate signer can be:

- Self - if the certificate signs itself, it is called a self-signed certificate. If you need to deploy a self-signed certificate, the certificate must be obtained from a secure channel. The only guarantee you have of the certificate's authenticity is that you obtained it from a trusted source.
- CA certificate - a more common solution is to sign certificates using a Certificate

Authority (CA) certificate. In this case, you only need to be careful about deploying the original CA certificate (that is, obtaining it through a secure channel). All of the certificates signed by this CA, on the other hand, can be distributed over insecure, public channels. The trusted CA can then be used to verify the signature on the certificates. In this case, the CA certificate is self-signed.

The user can use his own created CA or use any company offering CA services. The latter may be imposed by the company's guidance.

DirX Identity provides a set of batch scripts that allows you to create your own CA root and the key material for the Identity servers and client components. Note that for one machine, only one server private key is necessary (for all servers on the same host); the server's certificate is signed by the CA. Clients communicating with an ActiveMQ Message Broker only need one universal client key which is used on all hosts and need the CA certificate in their trust store in order to accept the server certificates issued by this CA.

The next sections describe how to set up and run these batch scripts.

##### 7.3.3.1.1. Configuring the Certificate Generation Scripts

Before you can run the scripts that generate the X.509 certificates, you must make the following updates to the batch script *install_path***ssl\set_Environment.bat**:

- Update the **dname** parameter. Make sure "localhost" is replaced by the local machine name. If you configured the servers with a non-qualified hostname, set the hostname as cn. If you configured the servers using a fully-qualified hostname (including domain name), set the same name as cn. Note that the name matching is case-sensitive. Use the same spelling when configuring DirX Identity and when setting up the scripts.
- Update the **validity** parameter. This parameter specifies the validity for all generated keys and certificates; the value is given in days. Note that on update or upgrade installations, the **set_Environment.bat** file is overwritten.
- Set the **pwd** environment variable in your environment (not in the batch script). The batch script uses this pass phrase for both the server key store and trust store. Make sure that you use the same password for all mentioned files in the configurator unless you have a distributed environment. Note that you will be prompted for the password for the CA private key.

##### 7.3.3.1.2. Setting up a Certificate Authority

To set up a certificate authority for a DirX Identity installation, run the SSL batch script:

*install_path***ssl\generate_CA.bat**



when you run this script, you will be asked for the pass phrase four times.

This script creates the following files:

- *install_path***ssl\ca.key** - the CA private key
- *install_path***ssl\ca.crt** - the self-signed CA certificate

The CA certificate file (**ca.crt**) and the files containing this certificate will be used on all machines in a distributed environment.



You only need to run this script once for a distributed DirX Identity installation.

#### 7.3.3.1.3. Setting up the Host Server Key

To set up a server key:

- If you are on a different host from where you set up the Certificate Authority, copy the **ca.key** and **ca.crt** files into the local *install_path**\ssl* folder.
- Run the script *install_path**\ssl\generate_ServerKeystore.bat*.

This script creates the following files:

- *install_path***ssl\identity-keystore** - the server's private key, in JKS format.
- *install_path***ssl\server.csr** - the server's certificate request (CSR), which needs to be signed by the CA.

#### 7.3.3.1.4. Signing the Server Certificate Request

The server's CSR file can be signed by an external CA or by the CA you created.

To sign the CSR file with your own CA:

- If you are on a host that is different from where you set up the CA, copy the **ca.key** and **ca.crt** files into the local *install_path***ssl** folder.
- Run *install_path***ssl\sign_ServerCSR.bat**.



You will be asked for the pass phrase one time.

This script creates the following file:

- *install_path***ssl\server.crt** - the server's certificate signed by the CA.

#### 7.3.3.1.5. Importing the Server's Certificate into the Shared Key Store

To import an external CA certificate or your own CA certificate and the generated server certificate into the shared key store, run the batch script:

*install_path***ssl\import_ServerCertKeystore.bat**

This script updates the following files:

- *install_path***ssl\identity-keystore** - the updated key store with both certificates
- *install_path***ssl\identity-truststore** - contains the CA certificate

If you obtain the CA certificate in DER format from the CA authority, you must adapt the import script. If you obtain an additional intermediate CA certificate, you must import this

certificate into the identity-keystore and identity-truststore, too.

##### 7.3.3.1.6. Converting the Server Key to PEM

You need to convert the server key contained in the shared key store from JKS format to PEM format because the C++-based Server uses PEM format. To convert the key to PEM, run the batch script:

*install_path***ssl\convert_ServerKeystoreToPem.bat**

This script creates the following files:

- *install_path***ssl\server-key.pem** - the key store in PEM format with the host-specific server key.
- *install_path***ssl\ca-crt.pem** - the CA certificate in PEM format without password.

##### 7.3.3.1.7. Setting up the Client Key

To set up a shared client key:

- If you are on a different host from where you set up the Certificate Authority, copy the **ca.key**, **ca.crt** and **identity-keystore** files into the local *install_path***ssl** folder.
- Run the script *install_path***ssl\generate_ClientKeystore.bat**. This script updates or creates the following files:

*install_path***ssl\identity-keystore** - the added shared client's private key, in JKS format.

*install_path***ssl\client.csr** - the client's certificate request (CSR), which needs to be signed by the CA.

##### 7.3.3.1.8. Signing the Client Certificate Request

The client's CSR file can be signed by an external CA or by the CA you created.

To sign the CSR file with your own CA:

- If you are on a host that is different from where you set up the CA, copy the **ca.key** and **ca.crt** files into the local *install_path***ssl** folder.
- Run *install_path***ssl\sign_ClientCSR.bat**.

Note: You will be asked for the pass phrase one time.

This script creates the following file:

*install_path***ssl\client.crt** - the shared client's certificate signed by the CA.

##### 7.3.3.1.9. Importing the Client Certificate into the Shared Key Store

To import the generated client certificate into the shared key store, run the batch script:

*install_path***ssl\import_ClientCertKeystore.bat**

This script updates the following file:

*install_path***ssl\identity-keystore** - the updated key store with the added client certificate.

#### 7.3.3.1.10. Converting the Client Key to PEM

You need to convert the client key contained in the shared key store from JKS format to PEM format because the C++-based Server uses PEM format. To convert the key to PEM, run the batch script:

*install_path***ssl\convert_ClientKeystoreToPem.bat**

This script creates the following files:

*install_path***ssl\client-key.pem** - the key store in PEM format with the client key.

#### 7.3.3.1.11. Importing the Certificate into the Java for DirX Identity

The CA certificate needs to be imported into the **cacerts** file of the Java for DirX Identity; that is, the Java that is used by the DirX Identity installation. You must perform this step on every machine where Java-based client/server components are installed; for example, DirX Identity Manager, Java-based Server, C++-based Server (Java-based agents), ActiveMQ Message Broker and so on.

To import the certificate with DirX Identity Manager:

- Start the DirX Identity Manager and log in.
- In the **Tools** menu, select **Options**.
- On the next page, the GUI's trust store is selected by default. (**This application's installation folder** is already selected and the file *install_path***GUI\cacerts** is displayed.)
- Select **Other**.
- Select the Java used by your DirX Identity installation. For example, **C:\Program Files\Java****jdkversion***\lib\security\cacerts*.
- Select **Import** and then select the CA certificate (*install_path***/ssl/ca.crt**) you want to import. When prompted for the key store password, enter the default value **changeit**.
- Click **OK**. The certificate is now imported.

To import the certificate with the **keytool** utility from the command prompt: (if there is no DirX Identity Manager installed on the machine):

- Open a command prompt.
- Run the command:

**keytool -import -noprompt -trustcacerts -alias sampleca -file** *install_path***/ssl/ca.crt -keystore** *dxi_java_home***/lib/security/cacerts -storepass** *password*

The default *password* for the **cacerts** file is **changeit**. The value for *dxi_java_home* is the path of the Java that was selected during DirX Identity product installation.

If you have a distributed environment, you must copy the following files from the **ssl** folder into the **ssl** folder on the target machine:

- If you installed a server (Java-based Server, C++-based Server, or ActiveMQ Message Broker), perform the steps described in "Setting up the Host Server Key", "Signing the Server Certificate Request", "Importing the Server's Certificate into the Shared Key Store" and "Converting the Server Key to PEM".
- If you installed a client component (Windows Password Listener, Web Center, Identity Manager, and so on), copy the files **identity-keystore**, **identity-truststore**, **ca-crt.pem**, **client-key.pem** and **password.properties** from a machine where a server is located into the local *install_path*/**ssl** folder.

#### 7.3.3.2. Securing the Identity Services

The following sections describe how to secure the Identity server and client components once the X.509 certificates have been generated.

Securing is done via setting a global SSL flag in the configuration.

To set the global SSL flags start the Initial Configuration Wizard again and choose the following four options:

- Connectivity Schema and Data Configuration
- ActiveMQ Message Broker Configuration
- C++-based Server Configuration
- Java-based Server Configuration

In the step **System-wide Configuration**, check the **Use SSL flag**.



In the steps for Message Broker, C++-based Server, and Java-based Server, make sure to specify for the hostname exactly the value you have specified inside the **set_Environment.bat** script described in "Setting up the X.509 Certificates". This is the full qualified domain name of the server.

##### 7.3.3.2.1. Securing DirX Identity Manager with SSL

To secure DirX Identity Manager with SSL, you only need to import the CA certificate into the Java on the machine on which DirX Identity Manager is installed, as described in "Importing the Certificate into the Java for DirX Identity". No further setup is required.

##### 7.3.3.2.2. Securing the Java-based Server with SSL

To secure the Java-based Identity Server with SSL, you only need to import the CA certificate into the Java on the machine on which the server is installed, as described in "Importing the Certificate into the Java for DirX Identity ". No further setup is required.

##### 7.3.3.2.3. Securing the C++-based Server with SSL

To secure the C++-based Identity Server with SSL, you only need to import the CA certificate

into the Java on the machine on which the server is installed, as described in "Importing the Certificate into the Java for DirX Identity". No further setup is required.

#### 7.3.3.2.4. Securing the Message Broker and Web Console with SSL

To secure the Message Broker with SSL, you only need to import the CA certificate into the Java on the machine on which the broker is installed, as described in "Importing the Certificate into the Java for DirX Identity". No further setup is required.

The ActiveMQ Web Console is also automatically configured for server-side SSL. You only need to import the CA certificate into your browser certificate settings. See next chapter for instructions for Firefox and Internet Explorer.

The URL to access it is by default **https://***myserver*:*port***/admin**, where *myserver* is the machine where your message broker runs and *port* is the Web Console port that you entered during initial configuration as the port.

You can configure the Web Console for client-side SSL access in the following steps:

- Configure the Web Console for client-side SSL:

  In the file *install_path***/messagebroker/conf/jetty.xml** in the section <bean id="handlers" class="org.eclipse.jetty.util.ssl.SslContextFactory">, add the following three lines:

  ```
  <property name="trustStorePath" value="install_path/ssl/identity-truststore" />
  <property name="trustStorePassword" value="your store password" />
  <property name="needClientAuth" value="true" />
  ```

- Create a p12 format truststore including the client private key and the client certificate. The installation provides a script that converts the identity-serverstore file into a p12 file. Note that only the identity-serverstore file includes the client private key. The script name is **convert ServerKeystoreToP12.bat/sh**.
- Include this p12 file to your browser's certificate store.
- Disable use of username/password authentication.

  The Web Console is configured to use username/password authentication which is unnecessary when using client-side SSL To disable it, in the file *install_path* /**messagebroker/conf/jetty.xml**, change the value from true to false in the line:

  ```
  <property name="authenticate" value="true" />
  ```

  This line occurs twice in **jetty.xml**: in the bean element with the ID securityConstraint and adminSecurityConstraint.

#### 7.3.3.2.5. Securing Web Admin and Server Admin with SSL

This section describes how to set up DirX Identity Web Admin or Server Admin for SSL to the Java-based Server. Both Web and Server Admin require access to the Java-based Server to handle HTTP / HTTPS requests for server management operations.

To set up Web Admin or Server Admin for SSL to the IdS-J server:

- Set up the following URL for the Web Admin: **https://***server:port***/admin**; for example, **https://***myserver:port*/**admin**, where *myserver* is the machine where your IdS-J server runs and *port* is the port that you entered during initial configuration as the port.
- Set up the following similar URL for the Server Admin: **https://***server:port*/**serverAdmin**
- On a Firefox browser, perform these steps:
- In Firefox, open **Options**→ **Options**. Select the Advanced tab.
- Click **View Certificates**. Open the **Authorities** tab and import the file **ca.crt**.
- On an Internet Explorer browser, import the CA certificate into the Microsoft certificate store:
- In Internet Explorer, Open **Tools**→ **Internet Options**.
- On the **Content** tab, open the **Certificates** dialog.
- Select the tab **Trusted Root Certification Authorities** and then click **Import**.
- The Certificate Import Wizard starts. Click **Next**. Browse to the file **ca.crt** in the **ssl** folder. Select **Trusted Root Certification Authorities** as the store in which to place the certificate. Click **Next** and import it. You must accept the security warning.
- An **Authentication Required** dialog requests a user name and a password (default: **admin** / **wE3!dirx**).

#### 7.3.3.2.6. Securing Web Center with SSL

This section describes how to set up DirX Identity Web Center for SSL to the Java-based Server or Message Broker. Web Center requires access to the Java-based Server to handle HTTP / HTTPS requests for request workflows. It requires access to the ActiveMQ Message Broker for sending events.

##### Managing Keys for Server-side SSL to the Java-based Server

The trust store for SSL from Web Center to the Java-based Server is either the file referenced by the Tomcat Java option **javax.net.ssl.trustStore** or the **cacerts** file of the Java used for Tomcat. If the Java option is set, the Java **cacerts** file is ignored. Note that any trust store configuration for Tomcat's HTTPS connector in Tomcat's **server.xml** file does not affect the SSL connections discussed here. You can use the same trust store here, but you still need to configure it separately via Java options.

When using a specific trust store, set the Java options for Tomcat if you haven't done so already.

Here is a Windows example:

- Activate the Tomcat Configuration Menu (**Start** → **Programs** → *Apache Tomcat version* → **Configure Tomcat**).
- Click the Java tab.
- Add **-Djavax.net.ssl.trustStore=***install_path***/ssl/identity-truststore** to the Java Options.
- (optional) Add **-Djavax.net.ssl.trustStorePassword=***truststore-password* to the Java Options. This step is optional. Reading certificates from a trust store usually works without a password; trust store integrity checks, however, are not performed.

Here is a UNIX example:

- Add **-Djavax.net.ssl.trustStore=***install_path***/ssl/identity-truststore** to your Tomcat environment variable **JAVA_OPTS**.
- (optional) Add **-Djavax.net.ssl.trustStorePassword=***truststore-password* to your Tomcat environment variable **JAVA_OPTS**. This step is optional. Reading certificates from a trust store usually works without a password; trust store integrity checks, however, are skipped.

Next, import the CA certificate *install_path*/ssl/ca.crt* into the trust store.

When using the Java **cacerts** file, import the CA certificate into the **cacerts** file of the Java you use for Tomcat:

- Start the DirX Identity Manager.
- In the **Tools** menu, select **Options**.
- On the next page, the GUI's trust store is selected by default. (**This application's installation folder** is already selected and the file *install_path***/GUI/cacerts** is displayed.)
- Select **Other**.
- Select the Java **cacerts** file used by Tomcat. For example, **C:\Program Files\Java***jdkversion***\jre\lib\security\cacerts**.
- Select **Import** and then select the CA certificate (*install_path***/ssl/ca.crt**) you want to import. When prompted for the key store password, enter the default value **changeit**.
- Click **OK** to import the certificate.

#### Managing Keys for Client-side SSL to the Message Broker

The **DIRXIDENTITY_INST_PATH** environment variable must be set in the context of the Tomcat process. The related files are:

- *install_path***/ssl/identity-keystore** - the key store for SSL from Web Center to the message broker.
- *install_path***/ssl/identity-truststore** - the trust store for SSL from Web Center to the message broker.
- *install_path***/ssl/password.properties** - the file from which the trust and key store passwords are obtained.

Check to make sure that the variable **DIRXIDENTITY_INST_PATH** exists. It can be missing

on some hosts in distributed environments with Web Center deployed into a native Tomcat. If the variable does not exist, continue with the following steps.

- Define the environment variable **DIRXIDENTITY_INST_PATH** using an existing directory folder as a value.
- Copy the files *install_path***/ssl/identity-keystore**, *install_path**/ssl/identity-truststore* and *install_path***/ssl/password.properties** to the directory folder you defined in **DIRXIDENTITY_INST_PATH**.

Note that the Java options **javax.net.ssl.trustStore** and **javax.net.ssl.trustStorePassword** are no longer supported for the Message Broker SSL.

The Java **cacerts** file is not evaluated here.

##### 7.3.3.2.7. Securing the Supervisor with SSL

Because the supervisor is deployed into the embedded Tomcat of the Java-based Server and the CA certificate is deployed in the Java **cacerts** file, no further action is required to secure this component.

##### 7.3.3.2.8. Securing Provisioning Web Services with SSL

This section describes how to set up the DirX Identity Web Services for SSL to the Java-based Server or the Message Broker. The Web Services require access to the Java-based Server to handle HTTP/SOAP or HTTPS/SOAP requests for request workflows.

If the Web Services have been deployed into the embedded Tomcat of the Java-based Server, no further action is required.

For Web Services deployed into a **native** Tomcat, you need to manage the keys for server-side SSL for communication with the Java-based Server and keys for client-side SSL for the Message Broker.

Check to make sure that the variable **DIRXIDENTITY_INST_PATH** exists. It can be missing on some hosts in distributed environments with Web Center deployed into a native Tomcat. If the variable does not exist, continue with the following steps.

- Define the environment variable **DIRXIDENTITY_INST_PATH** using an existing directory folder as a value.
- Copy the files *install_path***/ssl/identity-keystore**, *install_path***/ssl/identity-truststore** and *install_path***/ssl/password.properties** to the directory folder you defined in **DIRXIDENTITY_INST_PATH**.

If you have already configured SSL support on Tomcat, add the CA certificate *install_path***/ssl/ca.crt** to your Tomcat trust store.

If you have not used SSL support on Tomcat, do one of the following:

- Customize the Tomcat configuration:

On Windows:

- Activate the Tomcat Configuration Menu (**Start** → **Programs** → **Apache Tomcat** *version* → **Configure Tomcat**).
- Click the **Java** tab.
- Add **-Djavax.net.ssl.trustStore=***install_path***/ssl/identity-truststore -Djavax.net.ssl.trustStorePassword=***truststore_password* to the **Java Options**.

On UNIX:

- Add **-Djavax.net.ssl.trustStore=***install_path***/ssl/identity-truststore -Djavax.net.ssl.trustStorePassword=***truststore_password* to your Tomcat environment variable **JAVA_OPTS**.
- The preferred method is to import the CA certificate into the **cacerts** file of the Java you use for Tomcat.
- Start the DirX Identity Manager.
- In the **Tools** menu, select **Options**.
- On the next page, the GUI's trust store is selected by default. ("This application's installation folder" is already selected and the file *install_path***/GUI/cacerts** is shown.)
- Select **Other**.
- Select the Java **cacerts** file used by Tomcat (for example, **C:\Program Files\Java***jdkversion***\jre\lib\security\cacerts**).
- Select **Import** and then select the server certificate (*install_path***/ssl/server.crt**) you want to import. When prompted for the key store password, enter the default value **changeit**.
- Click **OK** and the certificate will be imported.

#### 7.3.3.2.9. Securing the Business User Interface with SSL

This topic is described in the *DirX Identity Business User Interface Configuration Guide* in the chapter "Configuring the Access to the DirX Identity REST Services".

#### 7.3.3.2.10. Securing the Windows Password Listener with SSL

To secure the Windows Password Listener (WPL) with SSL:

- Select the **useSSL** button in the Messaging Service dialog during the installation or set the property "UseSSL=1" after installation in the **options.ini** configuration file.
- Copy the files **ca-crt.pem**, **client-key.pem** and **password.properties** from a machine where they have been created into the local WPL *install_path***/ssl** folder.

#### 7.3.3.2.11. Securing the Meta Controller with SSL

To secure the meta controller (**metacp**) with SSL, you only need to import the CA certificate into the Java on the machine on which the meta controller is installed, as described in "Importing the Certificate into the Java for DirX Identity". No further setup is required.

#### 7.3.3.2.12. Securing JMX Clients with SSL

JMX clients require access to the Java-based Server to handle RMI requests for server management operations.

Example: To set up SSL from the Jconsole, call the Jconsole with the following command options (Windows):

**jconsole.exe -J-Djavax.net.ssl.trustStore="***install_path***/ssl/identity-truststore" -J-Djavax.net.ssl.trustStorePassword=changeme**

For more instructions on how to set up SSL connections for JMX clients, see the related Oracle documentation.

#### 7.3.3.2.13. Securing UNIX Scripts with SSL

This section describes how to set up UNIX scripts for SSL to the Java-based Server. UNIX scripts require HTTP / HTTPS access to the Java-based Server to handle requests to stop the server in a controlled way.

To prepare UNIX scripts for SSL, review the password settings in the file *install_path***/ids-j-***domain***-S***n***/clients/bin/shutdown_ini.sh**. They must match the passwords of the related key store and trust store.

## 7.3.4. Securing Connections between Web Services Clients and Web Services with SSL

Running the DirX Identity Web Services with SSL requires setting up the Web Services servlet and all clients that access it. However, client-side SSL is currently not supported for Web Services clients. The relevant tasks are

- Managing keys for server-side SSL
- Setting up SSL for Web Services clients for server-side SSL

### 7.3.4.1. Managing Keys for Server-Side SSL

Perform the following steps if Web Services have been deployed into a native Tomcat:

- Perform the steps described in the section "Securing Browser Connections with SSL", using the file *install_path***/ids-j-***domain***-S***n***/private/server-keystore** as the key store file.
- Add the server certificate into the trust store of the JRE used by the Web Services client. For the Web Service clients, this item is described in the *DirX Identity Integration Framework Guide* in the chapter "Web Services". For the Java-based Server, see the section "Sample Client". The relevant JRE is located in *dxi_java_home*.

These steps are required for setting up the Web Services client for client-side SSL.

#### 7.3.4.2. Managing Keys for Client-Side SSL

Client-side SSL is currently not supported for Web Services clients.

#### 7.3.4.3. Setting up SSL for Web Services Clients for Server-Side SSL

For the Web Services clients shipped with this installation, the configuration file *install_path***/provisioningServices/spmlv2/conf.xml** needs customizing.

Customizing is required for the **url** parameter of the definition block starting with

```
<connection type="Soap"
```

The **URL** parameter must be correct regarding the protocol (**http** or **https**), the host and port for addressing the Web Services.

Here are examples for correct URL specifications in the file *install_path***/provisioningServices/spmlv2/conf.xml**:

- To address Web Services deployed into the Java-based Server with secure port 40000: **url="https://myhost:40000/ProvisioningService-***technical-domain-name***/services/Spmlv2RequestService"**
- To address Web Services deployed into a native Tomcat with secure port 8443: **url="https://myhost:8443/ProvisioningService-***technical-domain-name***/services/Spmlv2RequestService"**

Web Service clients in general need customizing regarding the URL in their own configuration or software coding.

These steps are required for setting up the Web Services client for client-side SSL.

#### 7.3.4.4. Setting up SSL for Web Services Clients for Client-Side SSL

Client-side SSL is currently not supported for Web Services clients.

#### 7.3.4.5. Important Locations

Trust store: *java_install_path***/lib/security/cacerts**

Password for trust store: **changeit** (default)

This is the **cacerts** file of the Java environment used by your Web Services client.

### 7.3.5. Securing Connections to Tomcat Web Applications with SSL

The DirX Identity Web applications that run in an external Tomcat web server are:

- Web Center
- The Provisioning Web Service

- The Business User Interface
- The DirX Identity REST Service

Clients of these Web applications include:

- Browsers (for Web Center and the Business User Interface)
- The Business User Interface (for the DirX Identity REST Service)
- Custom clients (for the Provisioning Web Service or the DirX Identity REST Service)

To secure connections between the clients and the Web applications with server-side SSL, configure an HTTPS connector for Tomcat with an appropriate keystore in Tomcat's **conf/server.xml** file.

```
<!-- Example for a definition of an SSL HTTP/1.1 Connection on port
8443 -->
<Connector port="8443"
    protocol="org.apache.coyote.http11.Http11NioProtocol"
    maxThreads="150"
    SSLEnabled="true"
    keystorePass="changeme"
    keystoreFile="C:\Program Files\Atos\DirX Identity\ids-j-domain-
Sn\private\webcenter-keystore"
    clientAuth="false"
    sslProtocol="TLS"
/>
```

Adapt the parameters **keystorePass** and **keystoreFile**.The key in the Web Center keystore will do, but of course you can take any other appropriate key as well.



This configuration is part of Tomcat and may be subject to change.Check the Tomcat documentation if this hint does not work.

Add a certificate for the server's key to each client's certificate store (like a cacerts file for Java clients or the Trusted Root Certification Authorities store for Internet Explorer).You'll find a certificate for the Web Center key in the same folder as the Web Center keystore.

## 7.4. Understanding File-Handling Mechanisms

The DirX Identity runtime environment provides several mechanisms to handle files.Files can be:

- Configuration files, such as "ini" files, Tcl script files or attribute configuration files.
- Input and output files
- Trace or report files

The supported file-handling mechanisms are:

- Handling of absolute and relative paths
- Automatic procedures for file preservation
- Procedures for file transfers

### 7.4.1. Path Handling

There are two ways to set up file information in the configuration database:

- You can use relative paths (we recommend that you use only the short name of the file, for example "options.ini"). Relative paths allow high flexibility when the configuration must be changed or re-configured. All the DirX Identity examples in the initial database are configured this way.
- You can use absolute path names, but you should be aware of the consequences. DirX Identity cannot assist with absolute path name updates during reconfiguration of workflows or jobs. You should use absolute path names only when it is really necessary (for example, when you want to set up a central file pool as an interface to other applications).

### 7.4.2. File Preservation Mechanisms

DirX Identity provides the following mechanisms to prevent files from being overwritten on subsequent runs of an activity in a workflow:

- The C++-based Server does not allow you to start the same workflow simultaneously. An error message will be the result that is reported in the logging files and in the monitor view if the workflow was started via the scheduler.
- During a workflow run, the agent controller keeps all files that are relevant for a specific run of an activity in a specific work path folder (this works only if relative paths are used). This folder is defined by the work path in the corresponding C++-based Server object for this activity, the workflow display name, and the activity display name. The agent controller starts the agents in this (current) directory and assumes that an agent can handle any file location in the shared file system. If this is not the case, you must write a script when you configure of the job that moves and renames the necessary files (a pre- and post-wrapper).
- After the run of the activity, the agent controller moves all relevant files to a run-specific status area which is defined by the status path in the C++-based Server object, the workflow display name and start time, and the activity display name and start time. The agent controller writes the paths of these copied files to the corresponding status entry in the configuration database (you can define which files shall be copied by the **Copy to Status Area** flag in the file item object). The status tracker keeps the files until the expiration time of the corresponding status entry is reached. All other files remain in the work path (or at an absolute location when defined this way). You can define whether files should be preserved in the work path by the **Save Mode** flag in the file item object.

### 7.4.3. File Transfer Mechanisms

DirX Identity supports several mechanisms to transfer data files automatically from one location to the location at which this data is required. DirX Identity detects automatically whether the data resides on the local machine:

- If it resides on the local machine, links are used to access data with highest possible performance.
- If data is accessible via a shared file system (and this information is set up in DirX Identity) then data is simply linked and not copied.
- If data does not reside on the local machine and no shared file system has been set up (or this fact is not known to DirX Identity), then data is transferred automatically via the file service, which uses DirX Identity's messaging service. This data transfer is about one-third the performance of a shared file system (the reason is that two file copies must be performed). This is the default mechanism, which means that data is always transferred without any necessary setup configuration.

#### 7.4.3.1. Local Machine File Handling

To prevent interference between different activities of a workflow, each activity uses its own working directory. This mode of operation results in the following problem:

- The first activity works via its output channel into the related intermediate connected directory. For example:
  ...\work\myworkflow\A_activity\data.txt
- The second activity expects the data file in its input channel from the related intermediate connected directory. For example:
  ...\work\myworkflow\B_activity\data.txt
- Because the file is not in this directory, the problem must be solved.

DirX Identity automatically sets a link to the file in the working directory of the previous activity if both activities run on the same machine (specifically, it follows the channel to the intermediate connected directory that belongs to the previous job).

#### 7.4.3.2. Distributed Machines with Shared File Systems

To get maximum performance, you can set up a shared file system between two machines. Because DirX Identity cannot automatically detect this setup, you must configure it in the **Shared Paths** tab of the C++-based Server object.

DirX Identity automatically sets a link to the file in the working directory of the previous activity via the shared file system. This configuration guarantees the fastest possible performance because data is now transferred by the native operating system facilities.

If you have set up absolute paths for file handling and these paths are accessible via the shared file system, DirX Identity uses this mechanism.



Use a secure network connection if data files to be transferred contain critical data.

#### 7.4.3.3. Distributed Machines with No Shared File System Set Up

If a shared file system is not set up (or if it is not configured in DirX Identity), the file service copies the data from the source to the target machine using the messaging service's message queues. This mode of operation results in reduced performance (about one third compared with a shared file copy).

The messaging service transfer uses the set-up port (see the data port in the messaging service object in the configuration database). This port is the only port that must be configured in addition to the LDAP port to overcome a firewall.



You can optionally encrypt the file transfer mechanism.

For relative paths, the data file is copied from the working directory on the source machine to the working directory on the target machine.

If you have set up absolute paths for file handling, the file is copied from the fixed location on the source machine to the working directory on the target machine.

Please note that DirX Identity Manager uses the same mechanism to access files in the status area of all machines that belong to a DirX Identity domain.

# Appendix A: Context-Sensitive Help

This chapter presents the context-sensitive help topics that are provided with DirX Identity.

To display the help topic associated with the current dialog or menu, press **F1**.

## A.1. General

### A.1.1. Content

The actual content of a file, which may be a Tcl script, mapping function, INI file, XML file and so on.Use the **Export** button to export a file's contents from the configuration database into a file in the file system.Use the **Import** button to import the contents of a file in the file system into a contents object in the configuration database.

Related Topics

INI File
File Item
Mapping Function
Tcl Script
XML File

### A.1.2. Data File

Data files contain bulk data to be uploaded to or already downloaded from a connected directory.The respective configuration object holds all data necessary to identify and access the file.

Use this tab to assign the data file properties.The items shown in this tab are:

**Name**

the name of the data file configuration object.

**Description**

a description of the data file.

**Version**

the version of the data file.

**File Name**

the relative (short) name of the data file. This field can contain wildcards (specifically, regular expressions) which allows to handle a collection of files. You can also define a directory. Then all files in the directory are handled. Examples are:
trace*.trc - all trace files that contain a generated date.
*.rep - all report files with the extension 'rep'.
??data.dat - all files that start with two characters and end with 'data.dat'.
C:\myDataDirectory\ - all files that are contained in this directory.



Wildcard support depends on the specific agent that uses this file name specification. For example, the meta controller does not support wildcards. Check the corresponding agent documentation for more information.

**File Format**

the format of the data contained in an LDIF content file. This item is only used in connection with content type LDIF. It can take one of the following values:

**UNKNOWN**

a default value used for all files with a content different from LDIF.

**TAGGED**

the content of the file is tagged; that is, each item is in a separate row and written as *<name><separator char><value>*, for example Name:Miller.

**NON-TAGGED**

in an untagged data file, an individual attribute is identified based on its position in an entry. Attributes are separated by an attribute separator or a field width for the attribute can be defined. Example: ...|John|4375|Senior Developer|...

**LDIF-CHANGE**

a data file in LDIF change format, containing a list of directory modifications. Each entry in the change file contains a special LDIF "changetype" attribute that indicates the type of directory modification to be made. Example:

```
dn: cn=Joe Isuzu, ou=sales, o=Isuzu, c=us
changetype: delete
```

**FLAT-XML**

a simple XML format that contains objects and their attributes.

**DSML**

directory Service Markup Language (DSML) V1 format.

**Content Type**

the content type of the data file. This could be one of the following items:

**UNKNOWN**

an unknown or unspecified content.

**INI**

the data file contains configuration data in an INI file format.

**LDIF**

the content is structured in an LDAP directory interchange format.

**TCL**

the data file contains a Tcl script.

**XML**

the content of the data file is structured in XML format.

**Encoding**

the character encoding of the file. You can use any valid code set. See the *DirX Identity Meta Controller Reference* for details about code sets.
Note: By default, we use UTF-8 for all meta controller files and ISO-8859-1 for all other files (most agents cannot currently handle UTF-8).

***Keep Spaces**

by default, leading and trailing spaces are removed from all attributes. Setting this flag allows for keeping spaces.

**Save Mode**

the save mode of the data file within the workspace.It can take one of the following values:

**PERMANENT**

the file is stored permanently in the work area, which means it will exist even after the synchronization activity is finished.

**TEMPORARY**

the file will exist for as long as the synchronization activity is running.



Files that are needed as input for succeeding steps (activities) must be set to PERMANENT.

**Copy to Status Area**

whether or not the file is copied to the status area after the activity is finished.

**Related Topics**
INI File
Tcl Script
XML File

## A.1.3. Files

The Files tab lists all of the data files that comprise a connected directory.Use this tab to add files to or delete files from the connected directory.

**Data File**

the list of data files that comprise the connected directory.To insert or delete a file, use the respective button on the right.To display the properties of a data file, clicking on it, then click the **Details** button on the right.



If you create a new connected directory in the Global View, the Files tab will be empty during the copy procedure.The tab is filled in when you copy a workflow that uses this connected directory.The workflow copy procedure automatically creates all necessary data files in the connected directory

> and links the newly created channels to it.Do not create data files by hand during the connected directory copy procedure.

If DirX Identity were to copy all data files during the creation of the connected directory, a lot of unnecessary data files would be created that workflows would never use.

**Related Topics**

Connected Directory
File Item

### A.1.4. File Item

The properties of the indicated file, which may be a data file, a trace file, and so on.

**Name**

the name of the data file configuration object.

**Description**

the description of the data file.

**Version**

the version of the data file object.

**File Name**

the name or (optionally) path name of the file that must be recognized for status handling or generated before an agent is run by the agent controller.This field can contain wildcards (specifically, regular expressions) which allows handling a collection of files.You can also define a directory.Then all files in the directory are handled.Examples are:
trace*.trc - All trace files that contain a generated date.
*.rep - All report files with the extension 'rep'.
??data.dat - All files that start with two characters and end with 'data.dat'.
C:\myDataDirectory\ - All files that are contained in this directory.

**Content Type**

the content type of the data file. This could be one of the following items:

**UNKNOWN**

An unknown or unspecified content.

**INI**

the data file contains configuration data in an INI file format.

**LDIF**

the content is structured in an LDAP directory interchange format.

**TCL**

the data file contains a Tcl script.

**XML**

the content of the data file is structured in XML format.

**Encoding**

the character encoding of the file. Use any of the valid code sets. See the *DirX Identity Meta Controller Reference* for details.



By default, we use UTF-8 for all meta controller files and ISO-8859-1 for all other files (most agents cannot currently handle UTF-8).

**Save Mode**

the save mode of the data file within the workspace. It can take one of the following values:

**PERMANENT**

the file is stored permanently in the work area, which means it will exist even after the synchronization activity is finished.

**TEMPORARY**

the file will exist for as long as the synchronization workflow is running.

**Copy to Status Area**

whether (checked) or not (unchecked) the file is copied to the status area after the activity is finished.

**Related Topics**

Configuration Files
File Handling
INI File

## A.1.5. Query Folder

A query folder is a stored query that is used to filter a set of objects out of a much larger one. Use this tab to specify the filter criterion.

**Name**

the name of the folder.

**Description**

the description of the folder.

**Version**

the version number of the folder.

**Search Scope**

the scope for the search operation. Specify one of the following values:

**0-BASE OBJECT**

the search operation is performed on its start point only.

**1-ONE LEVEL**

the search operation extends to the start point and all objects that have the start point as their parent.

**2-SUBTREE**

the search operation extends to the whole subtree below its start point.

**Search Filter**

the search criterion in LDAP syntax; for example:
`"(&(dxmResult=closed.completed.ok)(objectclass=dxmWorkflowStatusData))"`

The following expression types can be used in the filter for time attributes:

- **$***base* or **$(***base***)** - represents the current time, depending on *base*. *base* can be:

  **NOW** or **gmtime** or **time** - the current time in GMT.
  **localtime** - the current time in local time zone.
  **date** - the time of this day start in GMT.
  **localdate** - the time of this day start in the local time zone.

  Examples:

  **`dxrExpirationDate>=$NOW`** - retrieves all entries that will expire in future.

  **`&(dxrStartDate>=$(date))(dxrStartDate⇐$(time))`** - retrieves all entries that were activated today up to now.

- **$***base operation constant* or **$(***base operation constant***)** - the time plus or minus a constant. The format of *constant* is:

  ***n*y*n*M*n*d*n*h*n*m*n*s**

  where *n* is the number of time units. The time units are:

  **y** years
  **M** months
  **d** days
  **h** hours
  **m** minutes
  **s** seconds.

  The order of time units is fixed, but each unit is optional. For example:

  **`(dxrStartDate>=$(NOW-3h))`** - retrieves all entries that were created within the last three hours.

  **`(dxrExpirationDate⇐$(gmtime+1y6M))`** - retrieves all entries that expire in one and a

half year.

A number without a time unit indicates days.

- **$***base operation* **$***variable* or **$(***base operation* **$***variable***)** - the current time plus or minus a variable.The values of these variables are the values described above for constants; for example:

  **`(dxrStartDate>=$(NOW-$Delta))`** - each time the filter is evaluated (select it or use the refresh button to start the evaluation) the variable is displayed with the previously entered value.Change the value if necessary and click **OK**.

- **$***variable* - the specified value is used in the filter, for example:

  **`cn=$StartsWith`*** - selects all objects where **cn** starts with the specified value.Each time the filter is evaluated (select it or use the refresh button to start the evaluation) the variable is displayed with the previously entered value.Change the value if necessary and then click **OK**.

**Max Result**

the maximum number of entries to be returned.

## A.1.6. Specific Attributes

DirX Identity can extend objects with virtual or specific attributes.

You can find specific attributes in any tab of an object.The display of these attributes does not differ from that of regular LDAP attributes.

All specific attributes that are not displayed in any of the other tabs are visible in the Specific Attributes tab.You can use the Specific Attributes Editor to add, modify or delete specific attributes in that tab.

Attributes visible in the Specific Attributes tab are extensions of the object that are not yet described by the XML object description.

Please note that references can either refer to regular LDAP attributes or to specific attributes.Examples are:

<?Job@dxmDisplayName/> for a regular LDAP attribute.

<?Job@dxmSpecificAttributes(Trace)/> for a specific attribute either visible directly in the specific attributes tab or in one of the other tabs.

To learn more about references, see the chapter "Customizing Object References" in the *DirX Identity Customization Guide*.

**Related Topics**

Channels
(Central) Configuration
Connected Directory

Job
Workflows

"Using the Specific Attributes Editor" in the *DirX Identity User Interfaces Guide*

### A.1.7. Tcl Content

Use this tab to edit the content of a Tcl script file or a mapping function.A special editor is provided to make the creation or modification of the script or the function as simple as possible.However, basic knowledge about the Tcl language is necessary for all modifications.For usage information on the Tcl editor, see the help topic in the DirX Identity Manager online help.



There are Tcl content windows that allow viewing but not editing.This is the case for the mapping script, where the content is created automatically when you edit the Mapping Item tab.Therefore you should not change this created content (all your changes would be lost after each change of the Mapping Item tab).



Another reason for a non-editable content tab is that this script is located in the DirX Identity installation area.Therefore you can only see a copy for informational viewing at the user interface level.This content also cannot be edited because this would not be reflected in the script that is really used by the workflows.Never change these scripts, because they might be changed during each update of the DirX Identity software (installation or patch).If you want to change a piece of this information, copy the routine from the script and put it into your local user_hooks script.Modify it accordingly and it will replace the original function.

**Related Topics**

Mapping Function
Tcl Script

### A.1.8. XML Content

Use this tab to edit the content of an XML file.XML files have pure text content and can therefore be edited with any text editor.DirX Identity Manager provides a simple editor control for this purpose.The two buttons below the editor enable the user to either export or import XML files.

**Import text...**

click to import a text file which will then replace the current content of the XML file.A file browser dialog is shown to select the file to be imported.

**Export text...**

click to export the current content of the XML file into an external text file.A file browser dialog is shown to select the desired directory and to type in the name of the text file.

**Related Topics**

XML File

## A.1.9. XML File

XML files are used in DirX Identity to define the extensions to any configuration object by Extensible Markup Language (XML) items.Extensions are especially needed for connected directory and job descriptions as well as for wizard configurations.

Use this tab mainly to assign a name to the XML file object.The properties shown in this tab are:

**Name**

the name of the XML file.

**Description**

the description of the object.

**Version**

the version number of the XML file.



If the XML file defines a wizard, we recommend that the wizard name specify the types of directories at the endpoints of the workflow (for example, LDAP and FILE).Between these endpoint values, you can include other information (for example, LDAP-3step-FILE) to identify a more specific wizard.

**Related Topics**

(Central) Configuration
GUI
Extensions

## A.1.10. Object Descriptions

A folder for DirX Identity Manager object extension design.Use this tab to assign a name to this folder.

**Name**

the name of the folder.

**Description**

the description of the folder content.

**Related Topics**

(Central) Configuration
GUI
XML File

### A.1.11. Wizards

A folder for DirX Identity Manager wizard design.Use this tab mainly to assign a name to this folder.

**Name**

the name of the folder.

**Description**

the description of the folder.

**Related Topics**

(Central) Configuration
GUI
Wizards
XML File

## A.2. Agents

### A.2.1. Agent

An agent configuration object describes the configuration data associated with a particular agent.Use the agent configuration object to describe each agent that is present in your Identity environment.Agent configuration objects can describe customer-supplied agents and agents developed by other companies in addition to the agents supplied with DirX Identity.

An agent typically synchronizes only one type of connected directory, but the meta controller is an example of an agent that synchronizes multiple directory types (LDAP and file-based).

The path to the agent's executable can be an absolute pathname or simply the name of the agent's executable file.When the path is the file name, the operating system facilities based on the path variable are used to find the executable.

DirX Identity supports any type of executable; for example ***.exe**, ***.cmd** or ***.bat** executable files on Windows systems.

The path property can also specify a "wrapping" batch file that the DirX Identity runtime is to call to perform pre- or post-processing functions in addition to the agent's execution.Adding a wrapping batch file allows you to extend an agent's capabilities to work with all of DirX Identity's features (for example, to prepare delta handling or to provide a more meaningful exit code that is derived by examining the trace file result).Be careful when using batch files to extend an agent's functionality, because batch files are operating system-dependent.

Use this tab to establish an agent's properties.

**Name**

the name of the agent.

**Description**

the description of the agent.

**Version**

the version number of the agent.

**Wrapper required**

whether the agent needs an agent wrapper (checked) or not (unchecked).

**Executable**

the path of the agent executable. You can specify either a relative or absolute path.

As a rule, you should specify only the file name of the executable (for example, **metacp.exe**). This will start the agent in the work area and allow easy reconfiguration when changing the C++-based server or the work path. If you specify the executable without the extension (for example, **metacp**), the agent can run on both Windows and UNIX.

Using an absolute path starts the agent in the specified directory, making reconfiguration more error-prone.

**OK Status**

the agent exit codes that indicate error-free execution. DirX Identity assumes that all exit codes in this list represent an error-free run. You can use the **OK Status** property in the job configuration object that uses this agent to override the exit codes defined here. Use a semicolon (;) to separate multiple values in the list.

**Warning Status**

the agent exit codes that indicate execution with warnings. DirX Identity assumes that all exit codes in this list represent runs with warnings. DirX Identity reports the warnings indicated by these exit codes but does not abort the workflow. You can use the **Warning Status** property in the job configuration object that uses this agent to override the exit codes defined here. Several values in this list must be separated by a ';' character.

DirX Identity considers all other agent exit codes to represent an erroneous run and stops the workflow's execution.

If the **OK Status** and **Warning Status** properties of a job and the agent have no values, DirX Identity treats each exit code as error. Hence you must at least specify one of the agent's success exit codes - usually exit code 0 - to make DirX Identity treat it as success.

**Abort Execution Allowed**

whether or not DirX Identity should stop the agent's execution when an exception occurs (typically as the result of manually aborting a workflow or shutting down the C++ server). By default, DirX Identity does not stop agent execution because the operation kills the related agent process, which can destroy parts of the information or make it inconsistent. Check this field to stop the agent's execution on exception.

**Agent Type**

the agent's type (for example, NT, Exchange, Notes, and so on).The agent type corresponds to an agent type configuration object.You can select another agent type here.Perform **Reload Object Descriptors** afterwards or restart the DirX Identity Manager.This will change the display behavior of all related Job objects.

**Directory Types**

the connected directory types that the agent can handle.

**Download AttrConf**

whether (checked) or not (unchecked) to download attribute configuration files.The meta controller needs attribute configuration files to control its operation.Check this field if your agent is based on the meta controller or needs a download of these files for other reasons.

**Related Topics**

Connected Directory

### A.2.2. Agents

A folder for the agent configuration objects in the Connectivity configuration database.Use this tab to assign a name to the agent folder.

**Name**

the name of the folder.

**Description**

the description of the folder.

**Related Topic**

Agent

## A.3. Collections

### A.3.1. Collection

A collection is a powerful method for exchanging data between different instances of Connectivity databases.Typically, you use it to export object sets into your software configuration system or to transfer them from the development system to an integration or production system.The data is stored in LDIF file format.



the LDIF file format depends on the collection properties in the **dxi.cfg** file.See the *DirX Identity User Interfaces Guide* for more information about this file.

For more information about transporting data with collections, see the chapter "Transporting Data" in the *DirX Identity User Interfaces Guide*.

Use this tab to define a set of objects, subtrees or rule-based object collections to be exported.

**Name**
the name of the object.

**Description**
the description of the object.

**Version**
the version number of this object.

**Path**
the path to which the LDIF file is to be written. Use the file selector box to define the path.

**Objects**
the objects to be exported. During the export operation, this list of objects is exported to the LDIF file. Only the defined object is exported, no subtrees or linked objects. You can define the objects with an object selector box.

**Subtrees**
the subtrees to be exported. After the export of the object list, the listed subtrees are exported to the LDIF file. You can define the subtrees with an object selector box.

**Rule-based**
the rule specification to use when exporting objects. After the export of the subtree list, the listed objects are exported based on a rule specification. Set the rule link to a rule definition.

**Collections**
the collections to be exported. After the export of the rule-based list, the defined list of collections is exported into the defined file of this collection (see the **Path** specification).

**Related Topics**

Collection
Collections
Collection Rule
Collection Rules

"Using Collections" in the *DirX Identity User Interfaces Guide*.

# A.4. Collections

A folder for collection configuration objects. Use this tab to assign a name and a meaningful description to the collection folder.

**Name**
the name of the folder.

**Description**
the description of the folder.

**Related Topics**

Collection
Collections
Collection Rule
Collection Rules

"Using Collections" in the *DirX Identity User Interfaces Guide*.

## A.4.1. Collection Rule

A collection rule is an XML-based rule that defines the export rule for the rule-based tab in the collection object.

Use this tab to define a set of objects and subtrees to be exported.

**Name**
the name of the object.

**Description**
the description of the object.

**Version**
the version number of this object.

**Content**
the rule definition in XML format.

A collection rule is to be defined in XML format, for example:

```
<rule>
<entry classes="dxmWorkflow" childLevel="1">
<link attribute="dxmActivity-DN" />
</entry>
<entry classes="dxmActivity" childLevel="1">
<link attribute="dxmRunObject-DN" />
</entry>
...
</rule>
```

This example exports a Tcl-based workflow object, follows the links to the activities, exports

these items and then follows the links to the run objects (typically jobs or other workflow definitions).

The full syntax is:

```
<rule>
<entry classes="objectclasses" [filter="filter"]
[childLevel="childLevel"] [parentLevel="parentLevel"]
[action="action"] >
 [<matchFilter ...> ... </matchFilter>]
 [<childFilter ... > ... </childFilter>]
 [<parentFilter ...> ... </parentFilter>]
 [<link attribute="linkAttribute"/>]
 [<link attribute="linkAttribute">
    <entry classes="objectclass" [filter="filter"]
[childLevel="childLevel"] [parentLevel="parentLevel"]
[action="action"] >
      [<matchFilter ...> ... </matchFilter>]
      [<childFilter ... > ... </childFilter>]
      [<parentFilter ...> ... </parentFilter>]
    </entry>
    ...
 </link>]
...
</entry>
...
<!-- use this default entry to process objects that are not yet
matched by previous rules -->
</entry classes="*">
</rule>
```

with these sub elements:

***objectclasses***

a space or comma separated list of object classes that is used in the LDAP search to retrieve this type of objects.

***filter*** **(optional)**

if an LDAP filter is defined, only the objects that match the filter condition are exported.



Filter definitions must be enclosed in brackets, for example use "(cn=RoleCatalogue)" instead of "cn=RoleCatalogue".
Alternatively you can specify a **matchFilter** DSML filter clause.

***childLevel* (optional)**

the depth of the sub tree to be exported. Possible values are:*
ignore* - ignore this entry completely
**none** - ignore this entry but follow the links
**all** or **0** - the whole subtree
**1** - just this entry and no sub objects (default value).
**2** - this entry and one level of sub objects
**3** - this entry and two levels of sub objects
...

***ldapFilter***

exports all children down to the level where the LDAP filter condition is valid (this object and lower level objects are not exported).
Note that filter definitions must be enclosed in brackets, for example use "(cn=RoleCatalogue)" instead of "cn=RoleCatalogue".
Alternatively you can specify a **childFilter** DSML filter clause.

***parentLevel* (optional)**

the level of parent objects to be exported. Possible values are:*
none -* no parents (default)
**all** or **0** - all parents
**1** - one level of parents above the given entry
**2** - two levels of parents above the given entry

i. +

***ldapFilter***

exports all parents up to the level where the LDAP filter condition is valid (this object and higher level objects are not exported).
Note that filter definitions must be enclosed in brackets, for example use "(cn=RoleCatalogue)" instead of "cn=RoleCatalogue".
Alternatively you can specify a **parentFilter** DSML filter clause.

***action* (optional)**

an action that defines how the entry is processed:
**default** - processes the entry (export or delete). This is the default.
**skip** - this entry is not exported or deleted but its child and parent definitions are processed.

***linkAttribute* (optional)**

the name of the attribute to be followed to other objects.
Use this syntax to define a specific attribute: "dxmSpecificAttributes:channelparent".

When processing an LDAP object (for example a user), then the entry elements are processed sequentially from top to bottom and the first matching element is used to process the object.



Entries can contain link definitions. Link definitions can itself contain entry definitions and so on. This allows defining a different behavior for the same object at different levels. It is also a means to control endless loops

effectively. The inner elements have higher priority than the root elements of the same type.

Hints for filter definitions

You can specify filter definitions in LDAP or DSML syntax.

- We recommend using LDAP filters because they are more compact and easier to read.
- Note that LDAP filter definitions must be enclosed in brackets. For example, use "(cn=RoleCatalogue)" instead of "cn=RoleCatalogue".
- If you need to specify values with special characters, for example '(' or ')', you have two options. For example, suppose you want to specify a value of 'abc(def)'
- Use LDAP filter escaping:
  (cn=abc\28def\29)
- Use a DSML filter:



```
<matchFilter
xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core">
<dsml:equalityMatch name="cn">
<dsml:value>abc(def)</dsml:value>
</dsml:equalityMatch>
</matchFilter>`
```

See also the delivered sample rules for sequentially more complex examples.

**Related Topics**

Collection
Collections
Collection Rule
Collection Rules

"Using Collections" in the *DirX Identity User Interfaces Guide*.

## A.4.2. Collection Rules

A folder for collection rule objects.Use this tab to assign a name and a meaningful description to the collection folder.

**Name**

the name of the folder.

**Description**

a description of the folder.

**Related Topics**

Collection
Collections
Collection Rule
Collection Rules

"Using Collections" in the *DirX Identity User Interfaces Guide*.

# A.5. (Central) Configuration

The (central) configuration folder object contains a number of global parameters that control the operations of the C++-based Server, workflow engine, agent controller, scheduler and status tracker components.Be careful when changing these parameters, because the changes you make can have a tremendous impact on DirX Identity runtime operation.

The subtree under the Central Configuration object contains other important objects for global configuration.



All of these objects are marked with a red border and the text "This object might be shared because it belongs to the Configuration folder".Be careful when editing such objects because this could affect other objects, too.

Use these tabs to set up the central configuration data.See the section "Basic Rules for Central Configuration Object Parameters" for correct setting of the following properties:

**Global Configuration**

**Name**

the name of the central configuration definition.

**Description**

descriptive text for this object.

**Version**

the version of this configuration entry.

**HA enabled**

whether (checked) or not (unchecked) high availability is enabled for this domain. Before checking this flag, make sure you set the other high availability parameters correctly, especially those for defining the monitoring circle and the ports for backup adaptors.

**SSL**

whether (checked) or not (unchecked) to secure connections between the Message Brokers in this domain and their JMS clients with SSL/TLS.

#### Proposals

central configuration lists that can be used in mapping functions or other Tcl scripts. Note: presently not in use.

### Server

#### Polling Time

the time between the checks for urgent requests such as aborts and keep-alives. The checks can be made by the C++-based Server or other components. The syntax format is *hh*:*mm*:*ss*. The default is 5 seconds.

#### Keep Alive Time

the time between the C++-based Server's checks on whether static threads like the scheduler and the status tracker are still running. If not, these components are automatically restarted. The default is 5 minutes. The wait time until a new component is started is calculated as 10 times the timeout defined in the dxmmsssvr.ini file (per default 30 seconds) plus the polling time (see above) which results per default to 10 x 30 +5 = 305 seconds (about 5 minutes). The syntax format is *hh*:*mm*:*ss*.

#### Latency Factor

the latency factor, expressed as a percentage. DirX Identity uses this value when calculating timeout values for workflows and jobs. Specifically, it multiplies a given timeout value with this factor. The default is 20%. Workflow timeout values are calculated as the sum of all job timeout values.

#### Thread Life Time

the time a thread can run in the C++-based Server. The default is 24 hours. Increase this value if you have workflows that run longer than 24 hours. The syntax format is *hh*:*mm*:*ss*.

#### Thread Cleaner Interval

the time between executions of the thread cleaner (part of the C++-based Server). The syntax format is *hh:mm:ss*. The default is 30 minutes.

#### Encryption Mode

the attributes to be encrypted:

##### None

No encryption

##### AdminPW

Administrative passwords are encrypted (passwords of DirX Identity's bind profiles that are needed to connect to the connected directories).

##### Attributes&AdminPW

Attributes and administrative passwords are encrypted.

### Status Tracker

**Status Life Time**

the default maximum time that status entries and related files will be retained following execution of a workflow. The syntax format is *hh:mm:ss*. The default is 1 month (720 hours). You can use the Status Life Time property of a workflow configuration object to set a workflow-specific status lifetime.

**Status Compression Mode**

Allows influencing the detail level and amount of status messages for all workflows (default configuration). This switch can help reducing load on the status tracker or simply avoid uninteresting status entries. These levels are available:

**0 - Detailed**

Detailed messages are sent during the workflow lifetime (compatibility mode, default)

**1 - Compressed**

Status messages are collected during the workflow run as far as possible and sent at the very end of a workflow. This reduces the number of status messages by 50 % or more.

**2 - Minimized if OK**

Only a workflow status entry is generated at the very end of a workflow if the workflow ends with status OK. No activity status entries are generated and no data is copied to the status area.

**3 - Suppressed if OK**

No status information is created at all and no data is copied to the status area if the workflow ends with status OK.

You can use this default switch and you can set this feature at each workflow entry individually.

**Start Time**

the start time for running the status tracker to delete status entries that have expired.

**Time Interval**

the time between executions of the status tracker to delete status entries. The syntax format is *hh:mm:ss*. The default is 24 hours.

**Deviation**

the maximum allowed deviation for running the status tracker to delete status entries. This is a plus range around the **Start Time**. The syntax format is *hh:mm:ss*. The default is 2 hours.

**Monitor only Provisioning Errors**

whether (checked) not (unchecked) monitor entries from the Java-based Provisioning workflows contain only error messages or other types of messages as well. By default, monitor entries for Java-based Provisioning workflows contain

messages like “INF(JOIN208): Object "cn=Alexander Gerber 5217,ou=accounts and groups,ou=Intranet,o=sample-ts" successfully modified.” in addition to potential error messages. Check this flag to suppress these types of messages and capture only error messages in monitor entries. As this flag is evaluated inside the Java-based Server, changing the flag does not immediately affect running Java-based Servers. A Load IdS-J configuration will propagate this change to the running Java-based Server. Internally the server periodically rereads/refreshes this flag every 30 minutes.

### Scheduler

#### Schedule Sync Interval

the interval between the scheduler’s checking of its schedule configuration objects. Changing a schedule object in the DirX Identity Manager forces the scheduler to reread the new schedule information. If this trigger mechanism does not work correctly, the scheduler rereads all schedules regularly at the interval specified here. The syntax format is *hh*:*mm*:*ss*. The default is 1 hour.

#### Disable Scheduling

enables/disables scheduling at all connected C++-based servers. You can also switch this flag with the **Disable/Enable Scheduling** menu at the **Schedules** folder.

### Specific Attributes

This tab allows you to set global parameters that are valid for all scenarios.

### Sub-Folders

The central configuration object contains the following subfolders:

Agent Types
Connected Directory Types
Connector Types
DirX Identity Servers
GUI
JavaScripts
Messaging Services
Notifications
Resource Families
Services
Standard Files
Supervisors
Systems
TCL
Topics

### Related Topics

C++-based Server
Specific Attributes
Status Handling
Workflows

## A.5.1. Agent Types

A folder for agent type configuration objects.Use this tab to assign a name to the agent type folder.

**Name**

the name of the folder.

**Description**

the description of the folder.

**Related Topics**

Agent Type
Agents
Agent
(Central) Configuration

### A.5.1.1. Agent Type

The Agent Types selection is a folder for the property descriptions of all the standard agent types supplied with DirX Identity.It allows for the extension of DirX Identity with new agents.The details of these types are described by a corresponding XML file (for example, the tabs and properties of the object) that is located in the folder **Object Descriptions**.

When you create a new agent type, it inherits the properties of one of the standard agent types.Consequently, you only need to describe in the XML file the differences between your new agent type and the standard one on which it is based.See chapter "Customizing Objects" in the *DirX Identity Customization Guide* for details.

The **Wizards** folder contains all wizards defined for this agent type.You can define additional wizards.

Because agents can have typical configuration files (for example, for import or export) you can place these files underneath the corresponding agent type configuration object.These central configuration files can then be referenced from any job, and edits to a central configuration file object apply to all jobs that reference the object.If your environment does not require this kind of centralization, you can keep the configuration files directly underneath the corresponding job configuration object.In this case, the files are set up as job-specific configuration files.

Use this tab to assign a name for the agent type.The properties shown in this tab are:

**Name**

the name of the agent type. This name must match the corresponding tag in the XML file.

**Description**

the description of the agent type.

An agent type configuration object can be refined by an optional XML file configuration

object. If an agent type does not have an associated XML file, it is represented as a generic agent type.

Agent type configuration objects can contain sub-objects; for example, Tcl script or "ini" file templates.

**Related Topics**

Agent Type
Agents
Agent
XML File

## A.5.2. Connected Directory Types

The Connected Directory Types selection is a folder for the property descriptions of all the standard directory types supplied with DirX Identity. It allows you to extend DirX Identity with new connected directory types. The details of these types are described by a corresponding XML file (for example, the tabs and properties of the object) that is located in the folder **Configuration Objects**.

Use this tab to assign a name to the connected directory type folder.

**Name**

the name of the folder.

**Description**

the description of the folder.

**Related Topics**

Connected Directory Type
Connected Directory
(Central) Configuration

### A.5.2.1. Connected Directory Type

The connected directory type configuration object defines a particular type of connected directory.The connected directory type object helps you to define new customer-specific directory types and integrate already existing synchronization solutions into a DirX Identity scenario with full control by the DirX Identity Manager.

When you create a new connected directory type, it inherits the properties of one of the standard connected directory types.Consequently, you only need to describe in the XML file the differences between your new connected directory type and the standard one on which it is based.See chapter "Customizing Objects" in the *DirX Identity Customization Guide* for details.

The **Wizards** folder contains all defined wizards for this connected directory type.Additional wizards can be defined by the customer.

Because connected directories can have typical information (for example, the attribute configuration information of a standard schema), you can place this information underneath the corresponding connected directory type configuration object.These central files can then be referenced from any connected directory, and edits to this central object apply to all of these connected directories that reference the object.If this centralization is not necessary, you can keep the files directly underneath the corresponding connected directory configuration object.In this case, the files are set up as directory-specific attribute configuration files.

Use this tab mainly to assign a name to the connected directory type.

**Name**

the name of the connected directory type. This name determines the display behavior of all objects based on this type. For the related XML description see the **Object Descriptions** folder beyond this object.

**Description**

the description of the connected directory type.

**Type**

the type of the connected directory. This field is used by the meta controller to determine the channel type to be handled (see the conn_param(dir_type) references in the control.tcl script). Typical values the standard script can handle are File and LDAP.

A connected directory type configuration object can be refined by an optional XML configuration object. If a connected directory type does not have an associated XML file, it is represented as a generic connected directory object.

**Related Topics**

Connected Directory Type
[_connected_directories]
Connected Directory
Object Descriptions
XML File

Agent
Agent Types
Files

## A.5.3. Connector Types

The Connector Types folder collects the set of all connector types known to DirX Identity. There is a subfolder for each connector type. This allows you to extend DirX Identity with new connectors. Currently, the only object you need to supply for a new connector is an object description for the Set Password workflow. It is located in the folder **Object Descriptions** and contains the list of modifiable properties for the workflow configuration and the description how to present them in the Manager.

The **Wizards** folder contains all wizards defined for this connector type. This folder is

currently not used and is reserved for future extensions.

Use this tab to assign a name to the connector type folder.

**Name**
the name of the folder.

**Description**
the description of the folder.

**Related Topics**

Connector Type
(Central) Configuration

#### A.5.3.1. Connector Type

A connector type object defines a particular type of connector.With the help of the connector type configuration object, the user is able to define new customer-specific connector types and can thus integrate new custom connectors built with the Identity Integration Framework into DirX Identity.

Use this tab to assign the parameters for the connector type.The properties shown in this tab are:

**Name**
the name of the connector type.

**Description**
the description of the connector type.

**Version**
the version of the connector type.

**Programming Language**
the programming language used. This value restricts the servers where this connector can be deployed; C++ and C# type connectors can run within IdS-C, Java type connectors can only run within IdS-J.
Available values are:

**C++**
standard object-oriented C++ language

**C#**
Microsoft's C# language (not yet available)

**Java**
Oracle's Java programming language

**Shared Library**
: the shared library that implements this type of connector.

**Connected Directory Type**
: the type of connected directory the connector can handle.

Connector type configuration objects can contain sub-objects; for example an "ini" file template.

**Related Topics**

Connector Types
(Central) Configuration
INI File

## A.5.4. DirX Identity Servers

A folder for the DirX Identity server configuration objects in the configuration database under (Central) Configuration.

**Name**
: the name of the folder.

**Description**
: descriptive text for this object.

The folder contains all configured Java-based and C++-based Servers as sub-objects.

**Related Topics**

(Central) Configuration
C++-based Server
Java-based Server

### A.5.4.1. Java-based Server

#### A.5.4.1.1. Adaptor - General

Adaptor entries reside below Java-based Server (IdS-J) entries and represent an adaptor that reads events from an event source.

Use this tab to specify the following adaptor properties:

**Name**
: the name of the adaptor.

**Description**
: a description of the adaptor.

**Active (read-only)**

whether or not the adaptor is active.If set, it indicates that this adaptor is loaded into the Java-based Server.

**Subscription ID (read-only)**

the name of the adaptor. It must be unique within all the adaptors of the same Java-based Server. When subscribing to JMS topics, it helps to build a unique client identification together with the domain and IdS-J name.

**Wait before retry (ms)**

the wait time before a retry after a connection problem to the Java Messaging Service (JMS) has occurred (default: 30 seconds).

**Encoding (read-only)**

the encoding of the XML configuration.

**Topic**

the name of the queue from which the adaptor reads or the topic to which the adaptor subscribes. If it subscribes to a topic, this is the part that identifies the type of messages. The topic is built as follows: *domain*.**prefix**.*cluster*

**Broadcast interval (ConfigurationHandler only)**

the interval in minutes at which the JMS list is broadcast to all subscribers (PasswordListener).

**Related Topics**

Adaptor - Limits
Adaptor - Configuration
Java-Based Server - General
Manage Servers - Adaptors

#### A.5.4.1.2. Adaptor - Limits

Use this tab to specify the following properties:

**Purger**

removes already deleted events from the adaptor queue.

**Interval (ms)**

the interval between purger runs (default: 60 seconds).

**Time Limit (ms)**

the maximum time for a purger run (default: 15 seconds).

**Priority**

thread priority (default: 8).Value range is 1 (low) to 10 (high).

**Pending Requests**

defines upper and lower limits for the number of pending requests.

**High Water**

the maximum number of pending requests stored in this adaptor's workspace.If this limit is reached, the adaptor stops reading events from the JMS queue.

**Low Water**

the number of pending requests stored in this adaptor's workspace when the server starts reading events from the JMS queue again.

**Repository**

parameters for the adaptor-specific repository most-recently-used cache in memory.

**MRU Cache Capacity (read-only)**

units (requests) held in memory (default: 1500). The rest is kept in the file-based repository.

**MRU Segment Cache Capacity (read-only)**

cache for faster access to the segment files (default: 50).

**Related Topics**

Adaptor - General
Adaptor - Configuration

Java-Based Server - General

Manage Servers - Adaptors

#### A.5.4.1.3. Adaptor - Configuration

This tab displays the complete XML definition for this adaptor configuration that is read as part of the Java-based Server configuration during server startup.

You can see that many values in the XML definition are expressed as variables, for example:

```
<technicalDomain>$\{DN4ID(THIS)@dxmTechnicalDomain}</technicalDomain>
```

or

```
<interval>$\{DN4ID(THIS)@dxmSpecificAttributes(purgerinterval)}</inte
rval>
```

These variables are replaced by its values when the configuration is loaded. You can see the resolved XML definition in the server.xml file in the logs directory of the Java-based Server.

**Related Topics**

Adaptor - General
Adaptor - Limits

Java-Based Server - General

Manage Servers - Adaptors

#### A.5.4.1.4. Java-Based Server - General

The Java-based Server (IdS-J) provides a runtime environment for Java-based workflows running in a distributed environment.An instance of a Java-based Server must run on each machine in the Connectivity domain where activities and connectors shall run.

The DirX Identity Java-based Server configuration object describes the configuration information for one instance.The DirX Identity installation procedure automatically creates a Java-based Server configuration object when selected during installation and configuration on a machine.

Use the Java-based Server tab to specify the following properties:

**Name**
the name of the server.For the naming scheme, see the section "**Naming Schemes**".

**Description**
a description of the server.

**State**
the state of the server (read-only). Possible values are:

**STARTED**
the server was successfully started.

**STOPPED**
the server was intentionally stopped.

**Service**
the service object that specifies the IP address and port number of the DirX Identity server performed at runtime. To display its properties, click the Properties icon on the right.

**Scheduler**
whether (checked) or not (unchecked) the scheduler runs on this server. For each domain, exactly one server must host the scheduler for Java-based workflows.



When you select design mode , you can display the server's XML configuration in a Configuration tab. Changing the configuration requires extensive knowledge of the Java-based Server configuration. The configuration contains references to LDAP attributes that are resolved

during server startup (Load IdS-J Configuration does not reload this configuration!). The resolved XML configuration can be found and checked in *install_path***ids-j-***domain*-S*n***log\server.xml**.



If you change attributes in this tab, you must restart the Java-based Server. Running the "Load IdS-J Configuration' command is not sufficient because it only loads the workflow definitions. Server restart is also necessary if attributes of a related object (service, system or messaging service) are changed.

**Related Topics**

Java-based Server - Domain
Java-based Server - Connectors
Java-based Server - Repository
Java-based Server - Limits
Java-based Server - Resource Families
Java-based Server - Status and Auditing
Java-based Server - Configuration

Adaptor - General

Manage Servers - Adaptors

#### A.5.4.1.5. Java-based Server - Domain

Use this tab to define domain-specific parameters, including:

**Domain**

the name of the Provisioning domain on which this server operates.

**Request Workflow Timeout Check (read-only)**

whether (checked) or not (unchecked) this server hosts the Request Workflow Timeout Check job for the Provisioning domain.The Request Workflow Timeout Check (previously named Full Check) is a special job that regularly checks timeouts of request workflows and their activities.If it detects a timeout, it sends a request so that the workflow engine updates the workflow state and, for example, terminates the activity or workflow.The Request Workflow Timeout Check job must run on exactly one server per domain.

**Related Topics**

Java-based Server - Connectors
Java-based Server - Repository
Java-based Server - Limits
Java-based Server - Resource Families
Java-based Server - Status and Auditing
Java-based Server - Configuration

Adaptor - General

Manage Servers - Adaptors

#### A.5.4.1.6. Java-based Server - Connectors

Use this tab to display a list of connectors and the following information about them:

**Name**
the name of the connector.

**Version**
the version of the connector.

**Shared Library**
the shared library that implements this connector.

Select a connector and click the Properties button on the right to display the properties of this connector.



The list is populated and updated automatically during the DirX Identity configuration procedure. If you want to install your own connectors, you should create a corresponding Connector Type and include it into the table to document this fact.



Changing attributes in this tab requires a restart of the IdS-J server. Using the command "Load IdS-J Configuration" is not sufficient because it only loads the workflow definitions. Server restart is also necessary if attributes of a related object (service, system or messaging service) are changed.

**Related Topics**

Connector Types

Java-Based Server - General
Java-based Server - Domain
Java-based Server - Repository
Java-based Server - Limits
Java-based Server - Resource Families
Java-based Server - Status and Auditing
Java-based Server - Configuration

Adaptor - General

Manage Servers - Adaptors

#### A.5.4.1.7. Java-based Server - Repository

Use this tab to display and edit parameters for repository control and backup.

**Repository Control**

**Repository Folder**

the folder of the persistent Java-based Server repository.All persistent information (for example the dead letter queue and the adaptor repositories) are written to this folder.The default is *install_path***/ids-j/repository**.Specify a full path name.

**Backup**

You can define a regular synchronized backup for the Java-based Server and the DirX LDAP server.The repository content is saved to a configurable location in a consistent state (recoverable).To enable backup operation:

- Set the parameters in this tab that enable backup.
- Activate the real-time **Joint Backup** workflow.
- Specify a schedule.

Backup parameters include:

**Active**

enables/disables backup.

**Backup Folder**

the folder where the backups are stored. The default value is *install_path***/ids-j/backup**. The result is one zip file named *account-nnnn-YYYYMMDD-HHMMSS* **-JavaRepository.zip** for all Java-based Server repositories. Specify a fully-qualified existing path name. On Windows, you can use a shared network drive; but then the IdS-J service must run under a different account from the system account.

Note that during the backup procedure, all server activities are stopped.



Changing attributes in this tab requires a restart of the IdS-J server. Using the command "Load IdS-J Configuration" is not sufficient because it only loads the workflow definitions. Server restart is also necessary if attributes of a related object (service, system or messaging service) are changed.

**Related Topics**

Java-Based Server - General
Java-based Server - Domain
Java-based Server - Connectors
Java-based Server - Limits
Java-based Server - Resource Families
Java-based Server - Status and Auditing
Java-based Server - Configuration

Adaptor - General

Manage Servers - Adaptors

#### A.5.4.1.8. Java-based Server - Limits

Use this tab to display and edit parameters for tuning and optimization.

**Pending Requests**

The server reads external events via the configured adaptors and stores them in its persistent workspace.Worker tasks can produce additional internal events.This section allows you to specify a limit for the maximum number of events stored in the workspace.

**High Water**

the maximum number of events stored in the workspace after which the server should stop reading external events.

**Low Water**

the number of events stored in the workspace at which the server should start reading external events again.



the server checks this limit and the memory limit below. The first limit reached stops reading external events.

**Memory**

The Java-based Server is configured to use a specific amount of memory. On the other hand, it reads external events and produces internal ones. Use the fields in this section to specify when the Java-based Server should stop reading external events and when it should start reading them again.

**High Water**

the limit in % at which the Java-based Server stops reading external events.

**Low Water**

the limit in % at which the Java-based Server starts reading external events again.

**GC Delay**

the wait time in seconds after a garbage collection call. The garbage collector is called if an adaptor was suspended after reaching the high water limit. The garbage collector tries to free memory. If the low water mark is reached, the adaptor is re-activated.



the server checks this limit and the pending request limit. The first limit reached stops the server from reading external events.

**Disk Usage**

The Java-based Server is configured to check disk usage through logging and repository files. On the other hand, it reads external events and produces internal ones. Use the fields in this section to specify when the Java-based Server should stop reading external events and when it should start reading them again.

**High Water**

the limit (as a percentage (%)) at which the Java-based Server stops reading external events.

**Low Water**

the limit (as a percentage (%)) at which the Java-based Server starts reading external events again. At this point, warning messages are written to the server log file.



set both values to 100 if you do not want to run the disk usage check

**Batch Queue**

For better performance, the Java-based Server combines incoming requests to batch jobs that are processed by worker threads together. This procedure significantly reduces initialization per event.

**Queue Size (default 200)**

the maximum number of events per batch job. When this limit is reached, the batch job is closed for further processing.

**Timeout (default 1000)**

the maximum time in milliseconds to wait between incoming events unless the batch job is closed for further processing.

**Tomcat**

This part of the tab contains parameters of the embedded Apache Tomcat that you may need to adapt to your environment. Tomcat is used by the Web Admin, Server Admin and for request workflow handling.

**Maximum No. of Threads**

the maximum number of threads that are used by the embedded Tomcat container in the server (default: 14).

**Server-Specific Threads**

This part of the tab contains some parameters that are related to specific server threads.

**No. of WorkflowengineThreads**

the number of threads that are used for the workflow engine itself which starts and controls workflow activities (default: 2).

**No. of ResultEntriesThreads**

the number of threads that are used for processing the end result of realtime events (default: 2). If you have a high amount of realtime events in your Identity environment, then it would make sense to increase the number of result entries threads.



Changing attributes in this tab requires restarting the Java-based Server. Using the command "Load IdS-J Configuration" is not sufficient because it

only loads the workflow definitions. Restarting the server is also necessary if attributes of a related object (service, system, or messaging service) are changed.

**Related Topics**

Java-Based Server - General
Java-based Server - Domain
Java-based Server - Connectors
Java-based Server - Repository
Java-based Server - Resource Families
Java-based Server - Status and Auditing
Java-based Server - Configuration

Adaptor - General

Manage Servers - Adaptors

#### A.5.4.1.9. Java-based Server - Resource Families

Use this tab to configure the resource families for each Java-based Server (IdS-J).This mechanism allows for controlling the number of threads for specific purposes in this server.As for load balancing, a subset of the messages is forwarded from one IdS-J to another, a message can be processed on every IdS-J in the same domain.As a result, make sure that all the resource families needed for running provisioning and request workflows are available on every IdS-J. The number of threads per server and resource family will indirectly influence the message load a server receives: the more slowly it processes messages, the fewer messages it will receive.

**Available**

the list of available resource families that have not been assigned.
Note: You can change or extend this list under Configuration → Resource Families

**Selected**

the list of assigned resource families.For each resource family, you can define the number of threads that shall run within this server.

Use the up- and down-arrow buttons between the two panes to add or remove resource families.

**Related Topics**

Java-Based Server - General
Java-based Server - Domain
Java-based Server - Connectors
Java-based Server - Repository
Java-based Server - Limits
Java-based Server - Status and Auditing
Java-based Server - Configuration

Adaptor - General

#### A.5.4.1.10. Java-based Server - HA (High Availability)

Use this tab to display and edit parameters for High Availability:

**Automatic Monitoring (supervisor only)**

whether (checked) or not (unchecked) the supervisor is running.

**Monitor C++-based servers (supervisor only)**

whether (checked) or not (unchecked) the supervisor hosted by this Java-based server (running as a Java-based Server component) is responsible for automatically monitoring all the C++-based Servers.If one of these servers is not available, the monitoring supervisor moves its workflows, activities and the Status Tracker to another C++-based Server.Note that this flag is evaluated only if **Automatic Monitoring** is checked.Set this flag for only one Java-based Server.

**Monitored Java-based Server (supervisor only)**

the Java-based Server to be monitored.When High Availability is enabled, the adaptor repositories of a monitored Java-based Server are backed up in real-time from the backup adaptor in a monitoring supervisor, which runs as a component in a Java-based Server.If the monitored server fails, the monitoring supervisor automatically (or an administrator manually using Server Admin) restores the messages from the repository backups and sends them to the Message Broker.

**Supervisor Configuration (supervisor only)**

the reference to the configuration entry for the supervisor.

#### A.5.4.1.11. Java-based Server - Status and Auditing

Use this tab to display and edit parameters for status handling and auditing.

**Auditing - General**

**Status Life Time**

the lifetime of status entries of statistic entries of successful performed workflows in the monitoring area of the Connectivity database.

**Status Life Time Error**

the lifetime of status entries of statistic entries of workflows that failed in the monitoring area of the Connectivity database.Usually this life time is longer than the life time of successfully performed workflows.

**Synchronous**

whether or not synchronous auditing is enforced.When set, the initiating workflow waits until the status entry / audit information is written to the status area or audit queue.When clear, auditing occurs asynchronously later on.If a lot of audit information is produced, it is possible that audit information is delayed for hours.Thus we recommend to run the server in synchronous mode.

**JMS-based Auditing**

whether or not audit XML files are generated. When clear, you need to provide the DirX Audit JMS plug-in that delivers audit messages via JMS directly to DirX Audit.

**Auditing - File-based**

**Audit Trail Folder**

the path where the audit trail information for real-time workflows is written. By default, the path is *install_path**\ids-j\logs*.

**Maximum No. of Records Per File**

the maximum number of messages an audit file can contain (default: **10000**). If this number is reached, the file is closed and a new file is opened.

**Auditing - JMS-based**

**Bind Profile**

a link to a bind profile holding a user name and password. We recommend creating a JMS bind profile at the corresponding Identity Store connected directory and setting the anchor to JMS. The specified user should have access rights sufficient for writing into the queue specified in JMS Queue Name.

**URL Message Broker**

the URL of the message broker that you have configured in the DirX Audit Configuration Wizard.

**JMS Queue Name**

the name of the message queue that you have configured in the DirX Audit Configuration Wizard for the DirX Identity JMS collector.

**Audit Trail Folder**

the directory in which the JMS plug-in stores the audit messages as temporary files when the JMS message broker is not available (one message per file).By default, the directory is local to the server.This is indicated by the placeholder ${IDM_HOME} which represents the home folder of the Java-based Server.If you specify a relative path, keep in mind that it is generated relative to the working folder of IdS-J (*dxi_install_path***/ids-j-***domain***-S***n***/bin**).

**Logging**

This section defines parameters for the logging files (server-**.txt or warning-**.txt). These parameters include:

**Logging Files Folder**

the fully-qualified path where the IdS-J writes its warning and server logging files. By default, the path is *install_path***\ids-j\logs**.

**Maximum No. of Records Per File**

the maximum number of messages a logging file can contain (default: **10000**). If this number is reached the file is closed and a new file is opened.

**Maximum No. of Files**

the maximum number of logging files in the logging directory (default: **100**). If this number is reached, the oldest file is deleted before a new one is opened.

**server.xml Dump Interval (sec)**

the interval at which the server writes to the server.xml file (default: 3600 seconds = 1 h). This file contains the complete configuration information updated at the selected interval. This information is important for analysis if the server encounters problems.

**Maximum No. of Overview Files**

the maximum number of overview files (default: 500) written by the Web Admin into the path *install_path***ids-j\logs\overview**.

**Eventlog Configuration**

**Enable (default: true)**

enables/disables logging to the Windows event viewer.

**Level (default: Warning)**

the level of logging from Info to All.

**Syslog Configuration**

**Enable (default: true)**

enables/disables syslog information generation on UNIX platforms.

**Host**

the host where to send the syslog information.

**Port**

the port where to send the syslog information.

**Level (default: Warning)**

the level of logging from Info to All.



Changing attributes in this tab requires a restart of the IdS-J server. Using the command "Load IdS-J Configuration" is not sufficient because it only loads the workflow definitions. Server restart is also necessary if attributes of a related object (service, system or messaging service) are changed.

**Related Topics**

Java-Based Server - General
Java-based Server - Domain
Java-based Server - Connectors
Java-based Server - Repository
Java-based Server - Limits
Java-based Server - Resource Families
Java-based Server - Configuration

Adaptor - General

Manage Servers - Adaptors

#### A.5.4.1.12. Java-based Server - Configuration

This tab displays the complete XML definition for this adaptor configuration that is read as part of the Java-based Server configuration during server startup.

You can see that many values in the XML definition are expressed as variables, for example:

```
<technicalDomain>$\{DN4ID(THIS)@dxmTechnicalDomain}</technicalDomain>
```

or

```
<interval>$\{DN4ID(THIS)@dxmSpecificAttributes(purgerinterval)}</inte
rval>
```

These variables are replaced by their values when the configuration is load. You can see the resolved XML definition in the **server.xml** file in the logs directory of the Java-based Server.



This tab is only visible if design mode  is selected in the Identity Manager toolbar.

**Related Topics**

Java-Based Server - General
Java-based Server - Domain
Java-based Server - Connectors
Java-based Server - Repository
Java-based Server - Limits
Java-based Server - Resource Families
Java-based Server - Status and Auditing

#### A.5.4.1.13. Domain for Identity Servers

This tab displays all common properties for the Java-based Servers of a domain.These properties include:

**Domain**

the domain name.

**Related Topics**

Java-Based Server - General
Java-based Server - HA (High Availability)

#### A.5.4.1.14. Manage Servers - Adaptors

Some of the JMS adaptors - called the permanent adaptors - can run in parallel on each Java-based Server.Others - the topic subscribers - must run on exactly one server per domain.You can disable a permanent adaptor on one or more Java-based Servers; for example, if you want to reduce load from one server.For the subscriber adaptors, you can select the server on which they should run.

- On the left side, you can see all installed Java-based Servers for the selected domain.
- On the top, you find the names of all available adaptors.
- In the middle, you can assign an adaptor to a specific Java-based Server or you can disable it.



If you assign an adaptor to a specific server, be sure that the corresponding workflows are active and loaded and that all necessary resource families are assigned to that server.

**Related Topics**

Adaptor - General

Java-Based Server - General

Manage Servers - Request Workflow Timeout Check

#### A.5.4.1.15. Manage Servers - Request Workflow Timeout Check

The Request Workflow Timeout Check component runs on exactly one Java-based Server per domain.This tab allows you to select this server.The left side of the tab displays the installed Java-based Servers.

**Related Topics**

Adaptor - General

Java-Based Server - General

Manage Servers - Adaptors

#### A.5.4.1.16. Manage Servers - Supervision

Each Java-based Server can monitor another Java-based Server in the same domain.

- The left side of the tab displays the installed Java-based Servers.
- The column "Automatic Supervision" shows whether supervision is active.
- The column "Supervised Server Name" shows the display name of the server to be monitored (an empty field means monitoring is not configured)

**Related Topics**

Adaptor - General

Java-Based Server - General

Manage Servers - Adaptors

#### A.5.4.1.17. Manage Servers - Schedule

The scheduler for Java workflows runs on exactly one Java-based Server per domain. This tab allows you to select this server. The left side of the tab displays the installed Java-based Servers.

**Related Topics**

Adaptor - General

Java-Based Server - General

Manage Servers - Adaptors

### A.5.4.2. C++-based Server

#### A.5.4.2.1. C++-based Server - General

The C++-based Server provides a runtime environment for the meta controller and the DirX Identity agents on distributed machines. An instance of a C++-based Server must run on each machine in the meta directory environment on which an agent is to run.

The C++-based Server configuration object describes the configuration information for one instance of a C++-based Server. The DirX Identity installation procedure automatically creates a C++-based Server configuration object when it installs a C++-based Server on a machine.

Use this tab to view and set the properties of a C++-based server:

**Name**
: the display name of the server.

**Description**
: a description of the server.

**Version**
: the version number of the server.

**Server Type**
: type of this server. Possible values are:

**Primary**
: each DirX Identity Connectivity domain must have a primary server (this is the first one that was installed). The primary server runs the Status Tracker by default.

**Secondary**

additional servers in a domain are marked with this value.

**Status Tracker**

enables/disables the Status Tracker component for this server. There should only be one server in with the Status Tracker is running. If you change this flag, you must restart the service.

During startup, the server checks whether the registered flag is set. If not, it simply registers. If it is set, checks are performed. The parameters in the INI file are checked against the parameters in the configuration database. If the parameters are correct, the server is registered. If not, the startup process is aborted. This mechanism prevents two different servers from registering at the same C++-based Server object (which could result in a lot of confusion).

C++-based Server startup is controlled by the **dxmmsssvr.ini** file in the path *install_path***DirX Identity\server**. All necessary startup parameters are defined there.

**Related Topics**

Service
C++-based Server - SOAP Listener
C++-based Server - SPML Receiver
C++-based Server - Configuration
C++-based Server - Paths
C++-based Server - Agents
C++-based Server - Agent Server State
C++-based Server - JMX Access
Standard Files

#### A.5.4.2.2. C++-based Server - SOAP Listener

Use this tab to view and edit the parameters of the C++-based SOAP listener.

**Soap Service**

the name of the related SOAP service. To display its properties, click the Properties icon to the right.

**Socket Accept Timeout**

the C++-based Server communication parameter that controls how quickly the server reacts to control events like configuration changes or requests to shut down. The value should be in the interval between approximately 100 and 1000 milliseconds.

**Socket Receive Timeout**

the C++-based Server communication parameter that defines how long the C++-based Server should wait for data sent to it via the SOAP interface. This value depends on the workload of the communicating partner, mostly the Java-based Server. If you experience "Receive timeout" errors, you should increase the value.

**Thread Check Interval**

the interval in milliseconds that controls how often the C++-based Server adjusts the number of threads. This value should be approximately 1 to 3 seconds. In highly dynamic environments (where work load is changing quickly), set it to a lower value.

**Accept / Retry number**

the number of times the C++-based Server should try to accept a connection before it generates a fatal error.

**Related Topics**

C++-based Server - General
C++-based Server - SPML Receiver
C++-based Server - Configuration
C++-based Server - Paths
C++-based Server - Agents
C++-based Server - Agent Server State
C++-based Server - JMX Access

#### A.5.4.2.3. C++-based Server - SPML Receiver

Use this tab to view and edit the parameters of the C++-based SPML receiver.

**URL Prefix**

the prefix of the URL of the SOAP service (default: **dxm.idsc**).

**Min. Number Receiver Threads**

the minimum number of threads in the C++-based Server that receive SOAP connections and interpret its data.

**Max. Number Receiver Threads**

the maximum number of threads in the C++-based Server that receive SOAP connections and interpret its data. For ideal server performance, there should be one receiver thread for every configured connector thread. You only set the minimum and maximum values here; the actual number of threads is controlled automatically by the server depending on the current workload.

**Low Water Mark**

the low-water mark of the SPML receiver ICOM queue (currently not used).

**High Water Mark**

the maximum number of requests that the C++-based server can accept without pushing them to the connectors. If some connectors do not work, the server accepts up to this number of requests and then refuses further requests. When the connectors start working again, they process the queued requests. The queue is deleted when the server is restarted.

**ICOM Timeout**

the interval at which the C++-based Server checks its internal queues. This value should be approximately 150 milliseconds.

**Related Topics**

C++-based Server - General
C++-based Server - SOAP Listener
C++-based Server - Configuration
C++-based Server - Paths
C++-based Server - Agents
C++-based Server - Agent Server State
C++-based Server - JMX Access

#### A.5.4.2.4. C++-based Server - Configuration

Use this tab to manage services and their limits.

**Services:**

**Service**
: the service object that contains the IP address and port number of the DirX Identity server. To display its properties, click the Properties icon on the right.

**File Service:**

**Buffer Size**
: the buffer size used by the file service queue (in bytes). The configured size can be anything between 32 KB (32768) and 1 GB (1073741824), the default value is 1 MB (1048576). The size must be higher than the size of the biggest file transferred between two different C++-based Server instances.

**Encryption Mode**
: the encryption mode for the transferred files. Select from the following values:

  **NONE**
  : transfer in clear text

  **Scrambled**
  : transfer in scrambled format (not readable but no high security)

  **Encrypted**
  : transfer in encrypted format (not readable with high security)

**Limits:**

**Max number of threads**
: the maximum number of threads that can be running in parallel in the server. The default value is 512, and you can only decrease this number because it is the maximum.



The required number of threads for a workflow can be calculated as: 1 + *number_of_activities*. Thus a workflow with 2 activities needs 3 threads during runtime. If a workflow runs distributed, the threads are distributed accordingly as defined.

**KeyGet Timeout**

the wait time for the agent to get the decryption key from the DirX Identity server during startup (in seconds; the default value is 100). Set this time to a slightly higher value if the agent runs on a slow machine.

**Related Topics:**

C++-based Server - General
C++-based Server - SOAP Listener
C++-based Server - SPML Receiver
C++-based Server - Paths
C++-based Server - Agents
C++-based Server - Agent Server State
C++-based Server - JMX Access

#### A.5.4.2.5. C++-based Server - Paths

Use this tab to manage C++-based Server paths.

**Path Definitions**

**Installation Path**

the pathname of the server's installation directory. Do not change this field because it is handled automatically by the installation routine.

**Work Path**

the pathname of the server's current working directory. All agents run with relative paths use the server's current working directory.

**Status Path**

the pathname for the directory where status entry files produced by workflow runs are stored. On Windows, a UNC path can also be used.

**Shared Paths**

If you have set up shared file systems to connect your machines in the DirX Identity domain with high performance, you must configure the information here. This information is used by both the C++-based server and the DirX Identity Manager (if it resides on a machine where a C++-based Server exists - otherwise you must set up the information in the **bindprofile.xml** file).

Use this part of the tab to define mapping information between the local file path and the remote file path. For each shared file definition, set one line in the table:

**Server**

the server to which you have access via the shared file system.

**Shared Path**

the path through which the remote server file system is accessible.

**Target Path**

the path on the remote server that needs to be mapped.

*Example 4. Shared Paths*

A shared file system exists between the local machine and the remote machine with the C++-based Identity Server name **MyRemoteOne**. You can access the shared files via the path G:\ from your local machine. The path on the remote machine is C:\Program Files\Atos\DirX Identity\work. Then the parameters are:

Server = MyRemoteOne
SharedPath = G:\
TargetPath = C:\Program Files\Atos\DirX Identity\work

Let's assume the file C:\Program Files\Atos\DirX Identity\work\wf1\act1\data.ldif shall be accessed via workflow wf1 and activity act1. Then DirX Identity checks whether one of the target paths fits with the first part of the file name. If it does, the file name is changed to G:\wf1\act1\data.ldif. If no hit is found, the file is transferred to the work directory on the local machine ...\work\wf1\act2\data.ldif via the file service.

**Related Topics**

C++-based Server - General
C++-based Server - SOAP Listener
C++-based Server - SPML Receiver
C++-based Server - Configuration
C++-based Server - Agents
C++-based Server - Agent Server State
C++-based Server - JMX Access

Standard Files
Files

#### A.5.4.2.6. C++-based Server - Agents

Use this tab to list all agents that are installed on the same machine as the respective C++-based Server. The tab shows two tables:

**Agents**

lists all agents that are installed on the machine where this C++-based Server resides (these entries are links to the corresponding agent object). Use the buttons to the right of the table to add and delete entries.

**Versions**

lists all agents and their version numbers that are installed where this C++-based server resides. You should use this information for reference during debugging or when errors are encountered. Use the buttons to the right of the table to add and delete entries.



Both lists are filled and updated automatically during the DirX Identity

> installation procedure. If you want to install your own agents, you should edit both tables to document this action.

**Related Topic**

Agent
C++-based Server - General
C++-based Server - SOAP Listener
C++-based Server - SPML Receiver
C++-based Server - Configuration
C++-based Server - Paths
C++-based Server - Agent Server State
C++-based Server - JMX Access

#### A.5.4.2.7. C++-based Server - Agent Server State

Use this tab to view the status of the C++-based Server and set its state.

**Registered**

whether the server is running. Registered is checked and not registered is unchecked.

**Start Time**

the time stamp for the last activation of the server.

**End Time**

the time stamp for the last termination of the server. If this field is empty, the server is currently running.

**Disable Scheduling**

enables/disables scheduling of this server. You can disables scheduling of all C++-based Servers with the Disable Scheduling flag at the central configuration object.

**Related Topics**

C++-based Server - General
C++-based Server - SOAP Listener
C++-based Server - SPML Receiver
C++-based Server - Configuration
C++-based Server - Paths
C++-based Server - Agents
C++-based Server - JMX Access

#### A.5.4.2.8. C++-based Server - JMX Access

Use this tab to view and edit the parameters of the C++-based server JMX access.

**JMX Service**

the name of the JMX access service. Click the Properties icon to display additional properties.

**JMX URL Prefix**

the prefix of the JMX Access URL.

**JMX Accept Timeout (ms)**

the C++-based Server communication parameter that specifies how fast the server reacts to control events like configuration changes or requests to shut down. Specify a value between 100 and 1000 milliseconds.

**JMX Receive Timeout (ms)**

C++-based Server communication parameter that specifies how long the C++-based Server waits for receiving data over JMX. This value depends on the workload of the communication partner. Increase this value if "receive timeout" errors occur.

**Related Topics**

C++-based Server - General
C++-based Server - SOAP Listener
C++-based Server - SPML Receiver
C++-based Server - Configuration
C++-based Server - Paths
C++-based Server - Agents
C++-based Server - Agent Server State

#### A.5.4.2.9. C++-based Server - Connector

A connector object defines a connector. It is a sub-object of a C++-based Identity Server object (IdS-C). A connector represents an instance of a connector type definition.

Use this tab to assign the parameters for the connector. The properties shown in this tab are:

**Name**

the name of the connector.

**Description**

the description of the connector type.

**Version**

the version of this definition.

**Connector Type**

the corresponding connector type object. Allows running several connectors for one connected directory.

**Is Active**

activates/deactivates the connector. Changing this parameter requires restarting the corresponding server.

**Connected Directory**

the connected directory with which this connector works.

**Bind Profile**

the relevant bind profile of the linked connected directory to login.

**Minimum Number of Threads**

the minimum number of threads of this connector type that run on this C++-based Server to process the corresponding requests.

**Maximum Number of Threads**

the maximum number of threads of this connector type that can run on this C++-based Server to process the corresponding requests. Set this number higher when you await a high workload for the connector. Set maximum to 1 when you have some connector whose implementation is not thread-safe

**Low Water Mark**

low-water mark of the connector's internal queue.

**High Water Mark**

high-water mark of the connector's internal queue.

**ICOM Timeout**

the ICOM timeout in milliseconds.

**Logging Number**

the number of the serviceability logging component. The range can be between 1 and 10. Higher numbers define more detailed logging.

**Related Topics**

Connector Types
(Central) Configuration

#### A.5.4.2.10. Get Server State

You can request server state information by opening any of the C++-based Server objects (located in the Expert View). Right-click on the object and then select **Get Server State**.

A window opens that shows all objects and their state in this server (click **Details** to see all of the information). The window is refreshed automatically every 15 seconds. This window can stay open while you work with DirX Identity Manager in parallel (for example, you can start workflows and then watch the objects appear and disappear in the server state window).

To abort a workflow, right-click it in the list and then select **Abort workflow**. Note: aborting a workflow does not kill running agent processes when the **Abort Execution Allowed** flag is not set in the corresponding agent object.

**StartUpTime**

start time of the server and time the server is on-line (local time, difference to GMT is indicated).

The meaning of the columns in the window for each component is:

**Name**
the component name (for non-standard components, the DN).

**Type**
the component type (scheduler, status tracker, workflow engine, agent, and so on).

**InstID**
the unique instance ID.

**Start time**
the start time of the component (local time).

**Disabled**
whether or not the scheduler component is disabled.

**LastMsgType**
the last received message type (create, execute, ...).

**Ack**
the number of acknowledge messages sent.

**IdleTime**
the amount of time that the component has been waiting for requests.

**ActiveTime**
the amount of time that the component has been processing a request.

**LastMsgReturn**
the last acknowledge return code sent.

### A.5.5. GUI

A folder for extensions to DirX Identity Manager's graphical user interface.Use this tab mainly to assign a name to the folder.

**Name**
the name of the folder.

**Description**
the description of the folder.

**Related Topics**

(Central) Configuration

#### A.5.5.1. Extensions

A folder for DirX Identity Manager user interface extensions (virtual object extensions).

The properties shown in this tab are:

**Name**
the name of the folder.

**Description**
the description of the folder.

The files from the folder **extensions** are automatically downloaded only during startup of the DirX Identity Manager (file system folder: *install_path***Gui\conf\extensions**).So don't forget to restart the DirX Identity Manager when you change any of the files in this folder.

**Related Topics**

(Central) Configuration
GUI
XML File

#### A.5.5.2. Report

The report dialog shows in the upper list all reports that can be used for the selected object.Only one object can be selected at a time.The pre-configured reports are:

- **Generic** - outputs the report information in a generic format.All attributes of all contained objects are displayed.This report in XML format is especially useful for further processing.
- *object* - reports the objects in a similar format as shown at the user interface level.This report type in HTML format is targeted for documentation.You can include the result easily into your documents.

You can also define your own reports.For information about how to set up your own reports, see the section "Customizing Status Reports" in the chapter "Customizing Auditing" in the *DirX Identity Customization Guide*.

The options in the report dialog are:

- **Report List** - show the list of available pre-configured reports.
- **Search Base** - display the selected object. You can change it here.
- **Search Scope** - choose the search scope. Available options are:
  **BASE OBJECT** - the report will only contain the selected object.
  **ONE LEVEL** - the report will only contain the children of the selected object (the selected object is not contained!)
  **SUBTREE** - the report will contain the selected object and the whole set of children at any depth.



some reports cannot work with all options.

- **Type** - select the output format. Available options are:
  **XML** - pure XML format. Best suited for further processing with XSLT processors or report tools with XML interface.
  **HTML** - standard HTML format. Can be included easily into your documentation.
- **Output to Viewer** - whether (checked) or not (unchecked) the report result is displayed in a new window with a simple HTML or XML viewer.
- **Output File** - when Output to Viewer is not checked, you can select the output file name and location here.

Select **Run Report** to start the report or **Cancel** to exit the report dialog.



If you try to report a huge amount of data (for example you selected a lot of entries or you use very detailed reports) Java could run out of memory. Be sure to have enough memory (or increase the Java memory accordingly).

Here are some recommendations for running reports on objects:

- Reporting a job or connected directory object should not be a problem.
- Do not report a very complex object (like the complete default application scenario or a folder with a lot of workflows). Report only one workflow at a time.
- Do not use the internal viewer for a large amount of data. Use for example the Internet Explorer instead (but this tool also has restrictions regarding the maximum size of HTML files).

#### A.5.5.3. Report Definition

The **Report Definition** object contains the following items:

**Name**
: the display name of the report definition object

**Description**
: a text description of the report definition object

**Version**
: the version of this object

**Content Type**
: restricts the usage of this report definition. For example, selecting **Workflow** in this field means that this report can only be used for objects of type workflow.

**Content tab**
: the producer and consumer definitions in XML format. See the section "Customizing Status Reports" in the chapter "Customizing Auditing" in the *DirX Identity Customization Guide* for details.

**Format tab**
: the XSLT conversion rules. See the section "Customizing Status Reports" in the chapter

"Customizing Auditing" in the *DirX Identity Customization Guide* for details.

#### A.5.5.4. Report Properties

Report properties specify the behavior of the report agent.

**Search Base**

the distinguished name at which to start the search.

**Report Name**

the name of the report file.

**Report Output Format**

the format of the output. Possible values are:
XML - XML format
HTML - HTML format

**Related Topics**

Files
Content

#### A.5.5.5. Reports

A folder for DirX Identity Manager report definitions. Use this tab to assign a name to this folder.

**Name**

the name of the folder.

**Description**

the description of the folder content.

**Related Topics**

(Central) Configuration
GUI
Report

### A.5.6. JavaScripts

The JavaScripts folder stores Java scripts that can be called from all Java-based workflows. Create your own folders if you need to extend DirX Identity with your script extensions.

The scripts provided here are used to define the mapping for password change workflows. The statistics script defines the calculation of the statistics information for status entries.

Use this tab mainly to assign a name for the JavaScript folder. The properties shown in this tab are:

**Name**

the name of the folder.

**Description**

the description of the folder.

**Related Topics**

(Central) Configuration
JavaScript Content
JavaScript File

#### A.5.6.1. JavaScript Content

Use this tab to edit the content of a JavaScript file.JavaScript files have pure text content and can therefore be edited with any text editor.DirX Identity Manager provides a simple editor control for this purpose.The two buttons below the editor allow you to export or import JavaScript files.

**Import text...**

click to import a text file which will then replace the current content of the JavaScript file.A file browser dialog is shown to select the file to be imported.

**Export text...**

click to export the current content of the JavaScript file into an external text file.A file browser dialog is shown to selected the desired directory and to type in the name of the text file.

**Related Topics**

JavaScript File

#### A.5.6.2. JavaScript File

JavaScript files are used in DirX Identity to define JavaScript user hooks and programs.

Use this tab mainly to assign a name to the JavaScript file object.The properties shown in this tab are:

**Name**

the name of the JavaScript file.

**Description**

the description of this object.

**Version**

the version number of this object.

**Related Topics**

(Central) Configuration
JavaScript Content

## A.5.7. Messaging Services

A folder for the messaging service configuration objects in the configuration database under the folder configuration.Use this tab to name the folder.

**Name**

the name of the folder.

**Description**

the description of the folder.

For messaging, the Apache Active MQ is used.A list of all installed message brokers is given in this folder.

There are two installation options:

- Multiple message brokers sharing one database for persistent messages. Only one broker at a time is accessing the database and accepts client connections. The shared database has to be located on a sheared network drive.
- Single message broker installation. The database can either be located on local drive or shared network drive.

The clients get their connection information from LDAP, where the installation procedure has stored the necessary data. Therefore, only message brokers installed by the DirX Identity installation procedure can be accessed by the clients. They use a static list of brokers.

For High Availability and failover, the ActiveMQ operations are used. If the accessed broker fails, the next broker takes over. The decision of which broker is the next is determined by the fastest exclusive access to the database.

**Related Topics**

(Central) Configuration
Messaging Services

### A.5.7.1. Messaging Service - Failover Transport Options

By default, this field is empty and the default failover configuration is used. We strongly recommend that you do not change the configuration, as it affects broker-to-broker and client-to-broker communication. This field can be used for project-specific changes.

If you use the failover transport options, the syntax is as follows:

*type value*

where *type* and *value* are separated by a space character. Keep in mind that ActiveMQ requires this option to be specified in case-sensitive format.

Example:

```
maxReconnectAttempts 5
```

For details on the failover options see the Active MQ documentation:

https://activemq.apache.org/failover-transport-reference.html.

**Related Topics**

Messaging Services

#### A.5.7.2. Message Broker

The message broker object represents an ActiveMQ message broker. Use this tab to view and change the parameters of a message broker. Parameters include:

**Name**
: the name of the message broker used internally and as the operating system service name.

**Description**
: a description of the message broker.

**Message repository**
: the location of the database for persistent messages. In case of multiple brokers, this database has to be located and accessible on a shared network drive

**Service**
: the name of the service, including connection parameters like port number, SSL usage and client authentication

**Related Topics**

Messaging Services

#### A.5.7.3. Message Broker - Transport Options

By default, this field is empty and the default failover configuration is used. We strongly recommend that you do not change the configuration, as it affects broker-to-broker and client-to-broker communication. This field can be used for project-specific changes.

If you use the transport options, the syntax is as follows:

*type value*

where *type* and *value* are separated by a space character. Keep in mind that ActiveMQ requires the option to be specified in case-sensitive format.

Example:

```
wireFormat.maxInactivityDurationInitalDelay 30000
```

For details on the transport options see the Active MQ documentation:

https://activemq.apache.org/configuring-transports.html.

#### A.5.7.4. Status Tracker (Topic)

Messages covering monitoring information on Tcl-based workflows are published to the Status Tracker topic. The status tracker runs on exactly one C++-based Server, subscribes to this topic and stores the message information into the Monitor area of the Connectivity database.



This tab may not be visible depending on the messaging service in use.

The tab shows all the properties of the status tracker message topic. It is strongly recommended that you do not edit any of the fields shown here since this could damage the current installation!

**Prefix**

the prefix for the status tracker queue (for example, Dxm.statustracker).

**Subscriber Queue**

the subscriber queue for the status tracker queue (for example, Dxm.statustracker.subscription).

**Stream**

the stream for the status tracker queue (for example, Dxm.statustracker.STREAM).

**Related Topics**

Messaging Services

### A.5.8. Resource Families

A folder for DirX Identity resource family definitions.

Use this tab to assign a name to this folder.

**Name**

the name of the folder.

**Description**

the description of the folder content.

**Related Topics**

Resource Family

#### A.5.8.1. Resource Family

Resource families control the deployment of activities on Java-based Servers.You can use resource families to control the load distribution of certain workflow types between Java-based Servers.However, because Java workflows can be distributed over all Java-based Servers of one domain, you should make sure that you assign each relevant resource family to each Java-based Server.

On one side, activities must be associated with resource families - each activity requires a certain resource family.On the other side, Java-based Servers provide resource families.An activity can only be processed on servers that host the required resource family.

A resource family is an abstract entity and represents a set of inter-changeable resources of some type.Typically these are connected directories of some type (LDAP or ADS or ...).If you need to access several connected directories of the same type, use different instance-specific resource families (for example LDAP1, LDAP2, ...).Another example can be a specialized system that can be used for encryption or other time-consuming tasks.

Use this tab to enter the properties of a resource family object.The items shown in this tab are:

**Name**

the name of the resource family.



do not use blank spaces in the name!

**Description**

the description of the object.

When you assign a specific resource family to a Java-based Server, you can configure the number of threads that will be created for this resource family on this server.

**Related Topics**

Java-based Server - Connectors
Java-based Server - Resource Families
Resource Families

### A.5.9. Services

#### A.5.9.1. Service

The service configuration object describes the configuration information for a particular service.Use the service configuration object to describe the different services in use in the Identity domain.A service configuration object can provide full or partial information about the service, such as its location within the network (IP address and/or server name), port numbers and a link to the corresponding system object on which it runs.

**Name**

the display name of the service object.

**Description**

A description of the service.

**Version**

the version number of the service object.

**Server Name**

the server name, if the service name is not sufficient or needs to be different when several access methods to the service are necessary (for example, per LDAP, which requires the IP address and port, or per native API, which requires a specific server name).

**IP Address**

the TCP/IP address of the server. This can be a number such as 218.34.52.12 or a DNS name. Use of DNS names is recommended.
Note: Due to compatibility reasons, the batch type workflows use this field. This field is not used by the IdS-J server and its components (workflows etc.). Instead they use the corresponding field in the system object.

**Data Port**

the data port number of the service for connections that do not use SSL.

**SSL**

whether (checked) or not (unchecked) the service requires SSL. Set this flag if this service requires a secure SSL connection (in this case, the Secure Port field is used instead of the data port).

**Client Authentication**

whether (checked) or not (unchecked) the service requires client authentication.

**Secure Port**

the secure port number of the service. Use this field to set up SSL connections.

**User Name**

the user name used for authentication when connecting to the mail server. This field is only shown when a mail service is configured (the attribute **dxmSpecificAttributes(ismailservice)** is set to **true**).

**User Password**

the user password used for authentication when connecting to the mail server. This field is only shown when a mail service is configured (the attribute **dxmSpecificAttributes(ismailservice)** is set to **true**).

**System**

the system on which the service runs. To display its properties, click the Properties button on the right.

**Related Topics**

Messaging Services
C++-based Server
Connected Directory
System

#### A.5.9.2. Services

A folder for the service configuration objects in the configuration database under the folder configuration.

**Name**

the name of the folder.

**Description**

descriptive text for this object.

Within a property page, the content of this folder is shown as a pop-up list of a combo box:

**Service**

the service object currently used by the object whose properties are displayed.Use the arrow button to pop up the list of all available service objects in the service folder.Use the Properties button to display the properties of the currently selected service object.



Service folders can be used to reflect customer-specific structures.The connected directory copy mechanism will keep these structures for copied objects.

**Related Topic**

(Central) Configuration
Service

### A.5.10. Standard Files

This folder allows defining standard files that are handled by the IdS-C server by default.If an agent writes such a file (defined by its file name), the server handles it as if it were an individual definition for this agent.

**Name**

the name of the folder.

**Description**

descriptive text for this object.

You can define all types of files here, for example Notification files, INI files or Tcl scripts.

**Related Topics**

Files
INI File

## A.5.11. Supervisors

### A.5.11.1. Supervisor

The supervisor controls other DirX Identity components in a high-availability environment to detect problems and to inform administrators or perform an automatic fail-over procedure.The properties of a supervisor object are:

**Name**
: the name of the object.

**Description**
: the description of the object.

**Version**
: the version number of the object.

***Notification**
: the link to a notification object that allows sending e-mail to an administrator if problems occur.

**Mode**
: the mode the supervisor performs currently. Switch the mode to influence the supervisor operation.

  **Stop**
  : the supervisor performs a controlled stop as soon as possible and ends the supervisor program.

  **Suspend**
  : the supervisor performs a controlled stop as soon as possible and waits in a loop for further commands. Use this value if you intend to reconfigure parts of the active configuration. Switch to operate after re-configuration.

  **Operate**
  : the supervisor performs normal operation.

**Active Configuration**
: reference to the active configuration entry under the supervisor parent node. If **Mode** is set to **Operate**, this configuration is active and the supervisor controls its elements. Choose another configuration to reconfigure this one as active configuration.

**Automatic Failover**
: enables/disables automatic fail-over. If set, the supervisor performs automatic fail-over. If clear, the supervisor informs the administrator via e-mail about the problem.

**Initial Delay**

the amount of time the supervisor is to wait after startup or re-configuration before checking the components. This delay allows the components to start up and run correctly before the supervisor checks them.

**Sleep Time**

the interval to sleep after all checks.

**Wait Tolerance**

the time period to wait before failure is assumed.If a component does not respond within the wait tolerance, a failure is assumed.The supervisor informs the administrator via notification (it also starts to perform the automatic fail-over routine if **Automatic Failover** is activated).

**Related Topics**

C++-based Server
Supervisor
Supervisor - Configuration
Supervisor - Server

#### A.5.11.2. Supervisor - Configuration

Each subentry below the supervisor entry can contain a complete (possible) configuration in a high availability environment.It can contain several high availability server objects.Use this tab to set up a Tcl-based supervisor configuration or a Java-based supervisor configuration.

The properties of a supervisor configuration object are:

**Name**

the name of the object.

**Description**

the description of the object.

**Version**

the version number of the object.

**Status (Tcl-based supervisor configuration only)**

the status of this configuration entry. Possible values are:

**INACTIVE**

this configuration is currently not used

**ACTIVE**

this is the active configuration entry

**Monitoring Interval (Java-based supervisor configuration only)**

the time (in minutes) between consecutive monitoring.

**Retry count (Java-based supervisor configuration only)**

the number of retries after which a monitored server is considered to be down.

**Related Topics**

Supervisors
Supervisor
Supervisor - Server

#### A.5.11.3. Java Supervisor Mail

The supervisor runs as a component of a Java-based Server. Use this tab to configure the properties of supervisor e-mails.

**Service**

the link to the service object that contains the mail host address.

**Subject**

the external text file to which to export the current content of the JavaScript file.Clicking this button opens a file browser dialog.Select the desired directory and enter the name of the text file.

**From**

an e-mail address.

**To**

one or more e-mail addresses.

**CC**

zero or more e-mail addresses.

**Body**

the e-mail text.

#### A.5.11.4. Supervisor - Server

The supervisor configuration server objects link a C++-based server object with workflows to be run on it (when the configuration is active).They may also contain a link to the corresponding fail-over configuration if this server fails.The properties of a supervisor configuration server object are:

**Name**

the name of the object.

**Description**

the description of the object.

**Version**

the version number of the object.

**C++-based Server**

the C++-based Server that is used as a high availability server in this configuration.

**Workflows**

the list of workflows that are re-configured to this C++-based Server when this configuration is active.Note: The re-configuration does not support distribution of the workflows and activities to distributed servers.

**Fail-over Configuration**

the configuration to be used by the supervisor during automatic fail-over when this server fails.If no server link is present, the supervisor informs the administrator and goes to suspend mode.

**Related Topics**

Supervisors
Supervisor
Supervisor - Configuration

#### A.5.11.5. Supervisors

A folder for the supervisor configuration objects in the configuration database.

**Name**

the name of the folder.

**Description**

descriptive text for this object.

**Version**

version of this object.

**Related Topic**

Supervisor

### A.5.12. Systems

#### A.5.12.1. System

The system configuration object describes the configuration information for a particular host system.Use the system configuration object to describe the different hardware platforms in use in the Identity environment.

A system configuration object can provide the location within the network (IP address and/or server name) and it can keep additional information about the system's hardware characteristics (CPU type, hard drive and RAM sizes) and the operating system it runs for

documentation purposes.

The system configuration object has the following tabs:

**System**
: Mostly informational data.Use this tab to record in the Connectivity Configuration database the information about the systems in use at your site.Once you have created a system configuration object, you must maintain it yourself (DirX Identity does not maintain it for you).

**Resources**
: Contains the resource families this system can handle.

**Security**
: Defines security-relevant information; for example, the trust store and key store on this system.

The properties of the system object are:

**Name**
: the name of the system.

**Description**
: the description of the system.

**Version**
: the version number of the system.

**IP Address**
: the TCP/IP address of the server. This can be a number such as 218.34.52.12 or a DNS name. Use of DNS names is recommended.
Note: This field is only used by the IdS-J server and its components (workflows etc.). For compatibility reasons, the batch type workflows use the corresponding field in the service object.

**Operating System**
: the system's operating system (for example, Windows 2022 Server).

**CPU Type**
: the system's CPU type (for example, Intel Pentium IV).

**HD Size [MB]**
: the size of the system's hard disk (in megabytes).

**RAM Size [MB]**
: the size of the system's RAM (in megabytes).

**Resource Families**
: the resources this system can handle. See the resource family object for more

information.

**Key Store Path**

the path to the key store on this system. The default is:_ install_path_*/security/java/keystore*.

**Key Store Password**

the password to access key store defined in **Key Store Path**.

**Trust Store Path**

the path to the trust store on this system. The default is: *install_path***/security/java/truststore**.

**Trust Store Password**

the password to access the trust store specified in **Trust Store Path**.

**Related Topic**

Resource Family
Service

#### A.5.12.2. Systems

A folder for the system configuration objects in the configuration database under the folder configuration.

**Name**

the name of the folder.

**Description**

the description of the folder.

Within a property page, the content of this folder is shown as a pop-up list of a combo box:

**Service**

the system object currently used by the object whose properties are shown.Use the arrow button to pop up the list of all available system objects in the service folder.Use the properties button to display the properties of the currently selected system object.

**Related Topic**

(Central) Configuration
System

### A.5.13. Tcl

#### A.5.13.1. Mapping Function

A mapping function is used to convert an attribute or a set of attributes of a source directory entry into an attribute of a target directory entry.Usually, such mapping functions

perform just simple operations like character escaping (removal or replacement of non-alphanumerical characters.However, more complex functions construct distinguished names or superior directory tree nodes.

Use this tab mainly to assign a name to a mapping function.

**Name**

the name of the mapping function.Note that this name will be used by the mapping editor to construct the mapping item.This name must match the name used in the Tcl proc header.

**Description**

the description of the mapping function.

**Version**

the version number of the mapping function.

**Argument Count**

the number of arguments this mapping function will take (at least the minimum number for a **Variable Argument Count**). The mapping editor uses this value to insert as many rows as needed when the mapping function is selected for use in a particular mapping item.

**Variable Argument Count**

whether (checked) or not (unchecked) the argument count of the mapping function is variable. The mapping editor uses this flag to allow the user to add additional rows to the respective mapping item where this mapping function is used, or to delete superfluous rows down to the number given in **Argument Count**.

**Related Topics**

(Central) Configuration
Control Scripts
Mapping Functions
Mapping Scripts
Other Scripts
Profile Scripts
Tcl

"Using the Mapping Functions" in the *DirX Identity User Interfaces Guide*
"Using the Mapping Editor" in the *DirX Identity User Interfaces Guide*

### A.5.13.2. Mapping Functions

The Mapping Functions folder contains the pre-defined list of mapping functions delivered with DirX Identity in the **Default** folder.The mapping editor uses the contents of this folder to provide a set of predefined mapping functions when you use the **Mapping Function** column in the mapping editor provided with DirX Identity Manager.These functions can also be extended with customer-supplied functions in a parallel folder.

Use this tab to assign general properties to the mapping function folder.

**Name**
the name of the folder.

**Description**
the description of the folder.

**Related Topics**

(Central) Configuration
Control Scripts
Mapping Functions
Mapping Scripts
Other Scripts
Profile Scripts
Tcl Script

#### A.5.13.3. Tcl Folder

The Tcl folder stores Tcl files and procedures that are common to all meta controller jobs. Create your own folders if you need to extend DirX Identity with your script extensions.

Use this tab mainly to assign a name for the Tcl script folder. The properties shown in this tab are:

**Name**
the name of the folder.

**Description**
the description of the folder.

**Related Topics**

(Central) Configuration
Control Scripts
Mapping Functions
Mapping Scripts
Other Scripts
Profile Scripts

### A.5.14. Topics

The folder that contains the topic and queue names used for DirX Identity messaging.

**Name**
the name of the folder.

**Description** - descriptive text for this folder.

**Related Topics**

Topic

#### A.5.14.1. Topic

This configuration object describes a topic or queue used for DirX Identity messaging.

**Name**

the name of the topic or queue specification.This name is only for documentation.It is not used by the software.

**Description**

descriptive text for this topic.

**Topic Alias**

the alias name of the topic or queue exchanged with the Java-based Server.This is the name that is used by software components to look up the topic value.

**Topic Value**

the name of the topic or queue.JMS clients use this value to identify the topic or queue to which they are sending message or from which they are receiving messages.

For more information about topics, see the section "Understanding the Java Messaging Service" in the "Managing DirX Identity Servers" chapter.

**Related Topics**

Understanding the Java Messaging Service

# A.6. Connected Directories

## A.6.1. Attribute Configuration - Details

Use this tab to view and edit information about the attribute configuration associated with a connected directory. This is the information the meta controller needs to handle connected directories. For details, see the *DirX Identity Meta Controller Reference*.

The Details tab consists of two tabs - Attribute List and Global Info - and the following buttons for importing and exporting attribute configurations:

**Import CFG File**

click to select and import an attribute configuration file to replace the current configuration.

**Export CFG File**

click to export the current attribute configuration to a selected attribute configuration file.

**Attribute List**

Lists the name, abbreviation, prefix, suffix, encryption flag, multi-value separator, length,

and match rule for each attribute. Only the abbreviation is needed for LDAP directories. The other parameters are needed for other directory types, such as the File type.

**S**

whether or not the attribute is to be deleted during a schema update. This feature is helpful if you use one attribute configuration for both the connected directory and the intermediate connected directory (most DirX Identity default applications are configured this way). Check this field for all attributes that are only available in the intermediate file connected directory but not in the target connected directory. A schema update in the target connected directory will then preserve these attributes (example: ADS). For schema update details see "Using the Schema Displayer" in the chapter "Using DirX Identity Manager" in the *DirX Identity User Interfaces Guide*.

**Name**

the name of the attribute (normally set equal to the Abbreviation). In some cases, you can define a name mapping here. For example in the ODBC workflows the *Table.Attribute* name in the ODBC database is mapped to the *attribute* name in the intermediate file (for example HR.Department=DEP).

**Abbreviation**

the abbreviation of the attribute. For LDAP directories, this field is the LDAP name.

**Prefix**

the value that precedes the attribute value in the file.
Note: For XML files the prefix must be followed by a colon (for example: "telephoneNumber:")

**Suffix**

the value that follows the attribute value in the file.

**E**

whether or not the attribute is transferred in encrypted mode (for example, a password attribute) and must be decrypted by the agent before use.

**MV Separator**

the separator used for multi-valued attributes.

**Length**

the maximum length of the attribute. Zero stands for an infinite length.

**Match rule**

the match rule for this attribute.

**Global Info**

Displays global information that applies to all the attributes in the current configuration:

**Record separator**

one or more characters (or the octal representation of one or more characters) enclosed

in single quotation marks (' ') that the meta controller is to use to distinguish between entries in a connected directory data file.

**Field separator**

one or more characters (or the octal representation of one or more characters) enclosed in single quotation marks (' ') that the meta controller is to use to distinguish between attribute values in entries within a connected directory data file that uses a tagged format.

**Comment**

one or more characters (or the octal representation of one or more characters) enclosed in single quotation marks (' ') that the meta controller is to use to identify comment lines in an LDIF-formatted connected directory data file or any other connected directory data file.

**Skip lines**

an integer value that is greater than or equal to 0 that specifies the number of lines from the beginning of the file that the meta controller is to ignore when processing an import file.

**Continuation line**

one or more characters (or the octal representation of one or more characters) enclosed in single quotation marks (' ') that the meta controller is to use to identify continued lines in an LDIF-formatted connected directory data file or any other data file with continuation lines.

**Enclosing seq**

one or more characters (or the octal representation of one or more characters) enclosed in single quotation marks (' ') that the meta controller is to use to identify the start and end of an entry in a connected directory data file.

**Op code field**

the attribute within an LDIF change file that holds the LDIF change file operation code for an entry in the change file.

**Add mod field**

the attribute in an LDIF change file that represents the "add" attribute modification operation of an LDIF "modify object" change operation.

**Replace mod field**

the attribute in an LDIF change file that represents the "replace" attribute modification operation of an LDIF "modify object" change operation.

**Delete mod field**

the attribute within an LDIF change file that represents the "delete" attribute modification operation of an "LDIF "modify object" change operation.

**Prefix (Base-64)**

the prefix Base 64 field defines information that the meta controller is to use to identify a

base64-encoded LDIF attribute using the attribute's private prefix followed by this global prefix information. Enter one or more characters (or the octal representation of one or more characters). If this field is not set, the meta controller uses only each attribute's private prefix when it parses the data file.

**New superior field**

the attribute within an LDIF change file that represents the "new superior" parameter in an LDIF "modify DN" change operation.

**New RDN field**

the attribute within an LDIF change file that represents the "new RDN" parameter in an LDIF "modify DN" change operation.

**Mod RDN op code**

the keyword in an LDIF change file that represents the LDIF "modify RDN" operation.

**Del old RDN field**

the attribute within an LDIF change file that represents the "delete old RDN" parameter in an LDIF "modify DN" change operation.

**Mod DN op code**

the keyword in an LDIF change file that represents the LDIF "modify DN" operation.

**Add op code**

the keyword in an LDIF change file that represents an "add object" LDIF change operation.

**Mod op code**

the keyword in an LDIF change file that represents the LDIF "modify object" operation code.

**Delete op code**

the keyword in an LDIF change file that represents the LDIF "delete object" operation code.

**Mod separator**

the attribute in an LDIF change file that identifies the end of an attribute modification in an LDIF "modify object" change operation.

**Ignore empty value**

whether or not the meta controller returns empty attributes (attributes with no value) in the results of an **obj search** operation on a connected directory.



LDIF files can contain the string **version:1** at the beginning. DirX Identity determines LDIF files when all these fields are set to these values:
Add op code:add
Delete op code:delete
Mod op code:modify
Mod DN op code:moddn

Mod RDN op code:modrdn
When all fields are set correctly, the string version:1 is generated as first line into the LDIF file.

**Related Topics**

"Using the Attribute Configuration Editor" in the chapter "Using DirX Identity Manager" in the *DirX Identity User Interfaces Guide*
Schema - Object Classes and Attributes

"Attribute Configuration File Format" in *DirX Identity Meta Controller Reference*

## A.6.2. Attribute Configuration - General

When synchronization between two connected directories is performed, the data of the source directory is downloaded by the respective directory agent into an exchange file and imported into the target directory by another agent. One of these agents is the meta controller program, which performs additional tasks to convert the data by applying appropriate mapping rules and ensuring consistency of the exchanged data. These tasks require the understanding of the data semantics which in turn makes it necessary to have a description of the downloaded data. This description is provided by the attribute configuration. For more information about the content of an attribute configuration, see the Attribute Configuration - Details help topic, which describes the structure in detail.

Use this tab mainly to enter a name for the attribute configuration. The properties shown in this tab are:

**Name**
the name of the attribute configuration.

**Description**
the description of the attribute configuration.

**Version**
the version number of the attribute configuration.

**Related Topic**

Attribute Configuration - Details
File Item

## A.6.3. Attribute Configuration Template

Use this tab to enter the attribute configuration information according to the given parameters.

See "Using the Schema Displayer" and "Using the Attribute Configuration Editor" in the chapter "Using DirX Identity Manager" in the *DirX Identity User Interfaces Guide* for more information.

### A.6.4. Bind Profile

A bind profile is needed to authenticate during setup of a connection to a connected directory.Bind profiles keep the administrative passwords that can be encrypted.

Use this tab to enter the required data for authentication.The properties shown in this tab are:

**Name**

the name of the bind profile.

**Description**

the description of the bind profile.

**Version**

the version number of the bind profile.

**Bind Parameters**

**User**

the user name for accessing a connected directory.
For **IBM Notes**, the value for the field **User** must be the path and file name of the User's ID file. For example:
C:\IBM\Notes\admin.id

**Password**

the user password for accessing a connected directory. To change the password, click the icon on the right. A password dialog opens that requires you to enter the new password twice.

Alternatively, you can reset the password to a default value with the right-most button.



the password is set in two fields in the directory:

- First, it is stored in the **userPassword** attribute of the bind profile object. Be sure to configure the directory correctly to use one-way encryption for this field. This field is used to authenticate users to change their passwords correctly. Because of the one-way encryption, DirX Identity cannot use this field for synchronization purposes.
- Second, the value is stored in the **dxmPassword** attribute. This field can either contain a scrambled value (if **Encryption Mode** in Configuration entry → Server tab is set to **None** and **Disable Encryption** is not set) or an encrypted value (otherwise). DirX Identity does not allow you to set a password in clear text into this field. This field is used to authenticate against the target connected directory to perform the synchronization task (for example, to synchronize passwords).

**Authentication**

(for LDAP Connected Directories only) the type of authentication:

- **SIMPLE** authentication with user name and password
- **ANONYMOUS** authentication without user name and password

**Protocol**

(for LDAP Connected Directories only) the LDAP protocol version:

- **LDAPv2**
- **LDAPv3**

**Security Token**

(for Salesforce Connected Directories only) the automatically generated key that is added to the end of the password in order to log into Salesforce from an untrusted network.

**Security Parameters**

**Disable Encryption**

whether (checked) or not (unchecked) encrypted passwords are automatically unscrambled. Set this flag if DirX Identity is to unscramble the dxmPassword value automatically before it is written into a configuration file (a Tcl or an INI file) because the agent cannot unscramble the value itself. If **Encryption Mode** is not set to **None**, setting or resetting **Disable Encryption** requires you to re-enter the correct password.

**Use Encryption (SSL) (relevant only for Tcl-based ADS workflow types)**

whether (checked) or not (unchecked) to use SSL. Note: for realtime ADS workflows, the SSL flag setting in the workflow's join → ts port controls whether or not to use SSL.

**SSL Connection (relevant only for certain workflow types; for example, LDAP workflow types**

whether (checked) or not (unchecked) to use SSL.

**Client Authentication**

For LDAP workflow types, additional parameters for client-side SSL can be set. (Note that the flag **SSL Connection** must be set, too):

**Client Authentication**

whether (checked) or not (unchecked) to use client-side SSL.

**Path to Key Store File**

the file name of the file-based key store containing the public certificate/private key pair and the relevant CA certificates for this client certificate.

**Key Store Password**

the password for accessing the key store.

**Key Store Alias**

the alias name of the key store entry (optional).

**Path to Trust Store File**

the file name of the file-based trust store containing the LDAP server CA certificate.

**Trust Store Password**

the password for accessing the trust store.

**Anchor**

the text value that helps to select the correct bind profile during reference resolution. See Reference Descriptions for details.

There may be additional bind profile properties associated with a specific connected directory type. See the *DirX Identity Connectivity Reference* for details about these connected directory-specific properties.

**Related Topics**

Connected Directory

## A.6.5. Bind Profile Container

A folder for the bind profiles associated with the connected directory. Use this tab to assign a name to the folder.

**Name**

the name of the folder.

**Description**

descriptive text for this object.

**Related Topic**

Bind Profile

## A.6.6. Bind Profiles

A list of the bind profiles associated with the connected directory. Use this tab to add, delete, or modify a bind profile.

**Bind Profiles**

the associated bind profiles.

To edit a bind profile, select it in the list and click the Properties button on the right.

To add a new bind profile, click the first button on the right.

To delete a bind profile, select it and click the second button on the right.

Click the third button to display the distinguished names of the table entries in text format.

For an **HCL Notes** connected directory, you need to define a separate bind profile for each

certifier ID file (see the *DirX Identity Connectivity Reference* for details).

**Related Topic**

Bind Profile

### A.6.7. Channels

A folder for the channel configuration objects in the configuration database.Use this tab to assign a name to the channel folder.

**Name**

the name of the folder.

**Description**

the description of this folder.

**Related Topics**

Channels
Job
Connected Directory

### A.6.8. Connected Directories

A folder for the connected directory configuration objects in the configuration database. Use this tab to assign a name to the folder.

**Name**

the name of the folder.

**Description**

descriptive text for this folder.

**Related Topics**

Connected Directory
Channels

### A.6.9. Connected Directory

A connected directory is a single data store in an Identity environment that exchanges data with the directory service.The connected directory object describes the configuration information for a particular connected directory.

Connected directories can either be located in the connected directories folder or under a job configuration object in the jobs folder for a Tcl-based workflow (these are called intermediate connected directories because they are only used in the middle of a workflow to exchange data via files).

Connected directories can have other properties assigned to them, for example, a set of login accounts (bind profiles) that the workflows use to gain access to the connected directories.

The connected directory configuration object can also have additional properties that depend on its type (for example, an Active Directory domain name or the ODBC data source name).

The connected directories also contain the data to be synchronized.You can use the DirX Identity Manager to view the contents of LDAP and file-type directories.

Use the Connected Directory tab to enter general properties of the connected directory. The items shown in this tab are:

**Name**
: the name of the connected directory.

**Description**
: the description of the connected directory.

**Version**
: the version number of the connected directory.

**Service**
: the link to the logical or physical access parameters (TCP/IP addresses etc.).

**Directory Type**
: the connected directory's type (for example, LDAP, Notes, and so on). To display its properties, click the Properties button on the right. You can select another connected directory type here. Perform **Reload Object Descriptors** afterwards or restart the DirX Identity Manager. This action will change the display behavior of the connected directory object itself and all related Channel objects.

**Subtype**
: the subtype that can be used, for example, by scripts to modify the behavior accordingly (example: RACF).

**Attribute Configuration**
: The related schema description in a format that DirX Identity can use. To display its properties, click the **Properties** button to the right.

**Viewer Command**
: the command for viewing the content of the connected directory (for example, setting Notepad for a file or MS Access for an .mdb database). This command is used by the open command for the connected directory. For configuration hints, see description below.

**Wizard**
: the wizard that can configure the connected directory. DirX Identity calls the wizard by building the name **conndir-*_wizard_**.xml*. This file must be located in the wizard folder

under Configuration → GUI.

**Associated Server**

the Java-based Server(s) on which to run Provisioning workflows for this connected directory. For more details on this topic and how to activate this setting, see the section "Managing DirX Identity Servers" → "Distributed Deployments and Scalability" → "Separating Traffic for Selected Connected Systems" in this guide.

**Listeners per Target System**

the number of JMS listeners (threads) per Java-based Server that should process Provisioning workflows for a target system associated with this connected directory. By default, each Java-based Server registers one listener for the corresponding queue. Note that only one listener is created for password changes.

There may be additional properties associated with a specific connected directory (depending on the Directory Type property). For example:

- Bind profiles, when the connected directory uses authentication
- File information, when the directory is a file-type directory.
- Schema information, when the directory's schema can be read automatically by DirX Identity.
- Channels, which are links between jobs or activities in the connected directory

#### A.6.9.1. Configuring the Viewer Command

You can associate a viewer command with most of the connected directories. This viewer command is used when you perform the open command from the context menu.

Different configuration methods exist:

- Connected directories of type **LDAP** - define the server and base node that shall be opened in the Data View. Two formats are possible:

  *server*

  *server:base_node*

Set the *server* name to the name field of the server (open a top-level node in the Data View with the Server command to view this field).

For example:

```
PQR
PQR:ou=development
```



The connected directory type definition must support this type of viewer command. Go to the Connected Directory Type (for example, by clicking the respective link in the Connected Directory property page), expand the subfolder "Object Descriptions" and then edit the XML object (for example,

ads-conndir.xml) below. Select the "Content" property page and replace the action

```
<action class="siemens.dxm.actions.ActionRunApp" name="open"
parameter="cdViewer.cfg" />
```

with

```
<action class="siemens.dxm.actions.ActionOpenDataView"
parameter="$[dxmOpenCommand]" name="open" />
```

- Connected directories of type **File** - define the relevant file viewer program and a fixed or variable parameter:

*viewer path_and_filename*
*viewer $(reference)*

where *reference* is a relative reference starting from the connected directory.

For example:

notepad C:\MetahubData\data.ldif

notepad $(dxmFile-DN@dxmFileName)



If the connected directory contains more than one file, one of the files is opened by random method.

- Special connected directories - check the documentation of the relevant vendor for the correct viewer command.

Example:

"mmc c:\Windows\system32\lusrmgr.msc" for the user management on Windows

**Related Topics**

Bind Profile
Connected Directory Type
Files
Operational Attributes
Scenario
Service
Specific Attributes
Workflows

## A.6.10. Operational Attributes

A connected directory's operational attributes control its runtime behavior.The operational

attributes used by a particular connected directory depend on its type:

**Use Operational Attributes**

whether the scripts handle operational attributes (by default: dxmOprMaster, dxrStartDate, dxEndDate, dxrState) for directory entries (True) or whether custom routines must be created (False).Supply one of the following values:

**TRUE** - operational attributes shall be handled.

**FALSE** - operational attributes shall not be handled (define custom routines instead).



You can define other operational attributes in the **Initialize** user hook function.Set the opr(master) to opr(enddate) field values accordingly.

**Master Name**

the master name of this directory. It can be used to set the dxmOprMaster attribute of directory entries.

**Creation/Search Base**

the base node that can be referenced by the scripts as a starting point for all operations. DirX Identity works with 'ou=mthb,o=PQR' as the default.
Note: This field can contain references. See section "Customizing Object References" in the *DirX Identity Customization Guide* for more information.

**Tombstone Base**

the node to which deleted entries will be moved when **Deletion Mode** in the Entry Handling tab is set to "MOVE".
Note: This field can contain references. See the section "Customizing Object References" in the *DirX Identity Customization Guide* for more information.

**GUID Prefix**

the short prefix used to generate a local GUID together with the unique identifier (Local GUID Attribute) from this entry master. This value is used by the workflow that exports from this directory.

**Local GUID Attribute**

the attribute name that holds the unique identifier in this connected directory. This value is used to compose the local GUID together with the GUID prefix. Is used by workflow that exports from this directory.

**GUID Attribute**

the attribute that should be filled with the generated GUID value in this connected directory. This attribute is used by a workflow that imports into this directory.

**Object Classes**

the list of object classes that must be added when an entry is created in this connected directory. All workflows that import into this directory can use this list. The specific set to take is defined by the Object Class Collection attribute of the appropriate channel.

**Physical Deletion**

the delta time after which an entry must be physically deleted in this connected directory (only used if **Deletion Mode** in the Entry Handling tab is set to "MARK"). A purify workflow can use this value to calculate the physical deletion time (expiration date + physical deletion).

**Related Topics**

Bind Profile
Connected Directory
Connected Directory Type
Files
Specific Attributes

## A.6.11. Provisioning Attributes

A connected directory's provisioning attributes control its provisioning behavior. The provisioning attributes used by a particular connected directory depend on its type.

**Central Definitions**

**Domain**

the distinguished name of the Provisioning domain (for example, "cn=My-Company"). Only used at the Identity Store.

**User Base in Identity Store**

the location where the user tree resides (for example, "cn=Users,cn=My-Company"). Only used at the Identity Store.

**in Identity**

**Account Base**

the location of the account tree in the Identity Store. Specify the full distinguished name, for example
cn=Accounts,CN=MyAD,CN=TargetSystems,CN=My-Company

**Group Base**

the location of the group tree in the Identity Store. Specify the full distinguished name, for example
cn=Groups,CN=MyAD,CN=TargetSystems,CN=My-Company



these two values are specified at the connected system's connected directory because they reference different target systems.

**in Connected Systems**

**User Base**

the location of the accounts in the connected system, for example
ou=Accounts,ou=Extranet Portal,o=MySampleDomain

Note: the term "Account" is not always used in connected systems. Examples of synonyms are "Users", "Staff", "Login" etc.

**Group Base**

the location of the groups in the connected system, for example ou=Groups,ou=Extranet Portal,o=MySampleDomain
Note: the term "Group" is not always used in connected systems. Examples of synonyms are "Roles", "Profiles" etc.

**for ADS**

**Relative User Base**

the domain-relative location of the accounts in the Active Directory, for example OU=USusers

**Relative Group Base**

the domain-relative location of the groups in the Active Directory, for example OU=USgroup

**Domain**

the domain path that defines the base point for the **Relative User Base** and the **Relative Group Base**
DC=mcha,DC=sis,DC=munich,DC=atos,DC=de

**UPN Extension**

the suffix of the User Principle Logon Name consisting of all domain name components. @mcha.sis.mcha.atos.de

**for Notes**

**Admin Request Database**

the name of the administration request database used by "AdminP" process of the IBM Domino Server installation. The administration request database contains requests that are asynchronously processed by "AdminP"; for example, moving/deleting mail files or moving users from one organizational unit to another.

**Admin Request Author**

the administrator who has access rights to the Administration Requests database. Each administrator who uses the Administration Process to perform tasks must have the appropriate access rights and roles in the IBM® Domino® Directory (NAMES.NSF), secondary directories - if applicable, Administration Requests database (ADMIN4.NSF), and the Certification Log database (CERTLOG.NSF).
For the Administration Requests database, administrators must have "Author" access to admin4.nsf. Therefore "Admin Request Author" defines the administrator who has the appropriate rights to access "admin4.nsf".

**Group Member Limit**

the number of members that are stored in a group.

**Unique Org Unit Attribute**

the attribute type that should be used in the IBM Domino Directory to hold the unique organizational unit (the organizational unit is not part of the full name and is therefore not retrievable. Therefore, DirX Identity uses an additional attribute in the IBM Domino Directory so that a lookup for persons with a specific organizational unit can be done).

**Notes Attribute Syntaxes**

the predefined definitions for IBM Domino Server attribute syntaxes. The IBM Domino Server supports a lot of different attribute syntaxes in its databases. The IBM Notes connector is not able to handle all of them and therefore supports only a subset. It can handle the following types: TEXT (strings), NUMBER (numbers), DATE (date definitions).

Normally the IBM Notes Connector can independently determine the attribute syntax of the fields. As a fall-back solution (if that calculation fails), the IBM Notes Connector uses predefined definitions:

- Text fields - specifies all attributes that are of type TEXT
- Number fields - specifies all attributes that are of type NUMBER
- Date fields - specifies all attributes that are of type DATE

**for SAP EP UM**

**Synchronize Service Users**

whether or not to synchronize Service Users to/from SAP EP UM. By default, these users are not synchronized.

**for SharePoint**

**Account Name Attribute**

the LDAP attribute containing the Windows Account Name in the Active Directory Peer TS Account.

**Domain Name Attribute**

the LDAP attribute containing the Windows Domain Name in the Active Directory Peer TS Account.

**Delete Group Enabled**

whether or not a SharePoint group is deleted when deleted in DirX Identity. If set to true, deleting a SharePoint group in DirX Identity results in deleting the corresponding group in SharePoint.

**Filter Block Size**

the maximum number of members that are combined in one search filter. Searching the accounts in DirX Identity is done in combined searches for multiple members. The Filter Block Size parameter defines the maximum number of members that are combined to one search filter. If there are more members to be processed, additional searches are performed.

### A.6.12. SAP ECC UM Parameters

The SAP ECC UM connected directory defines the following additional parameters:

**CUA enabled**
: whether (checked) or not (unchecked) Central User Administration (CUA) is enabled on the SAP application system.

**Logon Variant with**
: how to connect to the connected SAP application depending on the SAP application system type. Possible values are:

**0 = No load balancing** - single application server.
**1 = No load balancing but via gateway** - over a gateway server.
**2 = With load balancing** - via a message server:

**System Number**
: the system number to be used for connecting to the ECC system (logon variant 0 or 1).

**Server**
: the host name or the IP address of the application server (logon variant 0 or 1) or the host name/ IP address of the message server (logon variant 2).

The host name and the service name of the application or message server must be defined in the hosts and services files:
Logon variant 0 or 1: *service name* = **sapdp***system_number*
Logon variant 2: *service name* = **sapms***ECC_system_name*

**Gateway Host**
: the host name or the IP address of the SAP gateway server (logon variant 1).

**Gateway Service Number**
: the gateway service number (logon variant 1). For example: **sapgw00**.

**R/3 System Name**
: the name (system ID) of the ECC system (logon variant 2).

**Group Name**
: the name of the group of application servers (logon variant 2).

If a Secure Network Connection (SNC) should be used, the following parameters are necessary:

**Mode**
: whether (checked) or not (unchecked) an SNC connection is used.

**Library**
: the full path and file name of the SAP Cryptographic library.

**Partner Name**

the application server's SNC name.

## A.6.13. Proxy Server

A connected directory's proxy server settings are used for central HTTP/HTTPS proxy configuration for Java-based Provisioning workflows.

**Proxy Server**

a link to the Proxy Server Connected Directory, where the host name of the Proxy Server must be specified in the linked service object.

**Proxy Server Bind Profile**

a link to the bind profile of the Proxy Server.The authenticated access to the Proxy Server is currently not supported.

## A.6.14. Schema - Object Classes and Attribute Types

The Schema configuration object shows the properties of a schema associated with a connected directory.It presents two tabs: Object Classes and Attribute Types.

**Object Classes**

This tab displays the names, IDs, and kinds of object classes after a schema read.

The first column allows you to define whether or not the object class should be used to update the attribute configuration.If this column is flagged, then all attributes of this object class are transferred to the corresponding attribute configuration object; if not, nothing is transferred.DirX Identity does not allow you to specify this information at an attribute level because you can do it with the **Selected Attributes** feature for each individual channel.For schema update details, see "Using the Schema Displayer" in the chapter "Using DirX Identity Manager" in the *DirX Identity User Interfaces Guide*.



The schema information is not stored in the configuration database.It is only kept in DirX Identity Manager's cache for the current session and therefore no longer available after a restart of the Manager.

To display the properties of a selected object class in the list, click the Properties button on the right. A dialog is opened showing the following properties:

**Name**

the name of the object class.

**Description**

the description for the object class.

**ID**

the unique identifier for the object class.

**Kind**
the type kind of the object class.

**Superior class**
the superior class of the object class.

**Obsolete**
whether (checked) or not (unchecked) the object class is obsolete.

**May and must attributes** - the list of optional and required attribute types. The meaning of the property items for an attribute type is described below. For a required attribute, the check box Mandatory is marked.

**Attribute Types**

This tab displays the properties of object attribute types in the schema. For each object attribute selected from the list, the tab displays its name, ID and length in a table and the remaining properties as fields in the tab. If you cannot see the table on a small screen, try to resize the vertical border of the pane upwards. The available fields are:

**Name**
the name of the attribute type.

**ID**
the unique identifier for the attribute type.

**Length**
the maximum value length for the attribute type.

**Description**
the description of the object attribute.

**Derived from**
the attribute from which this attribute is derived.

**Syntax**
the syntax for using this attribute.

**Usage**
how to use this attribute type.

**Match Rules**
the Equality, Ordering, and Substring match rules for the attribute.

**Options**
the selectable options for the attribute: Collective, Modifiable, Single value, and Obsolete.

**Include Orphaned Attributes**
whether (checked) or not (unchecked) to transfer all attributes that do not belong to any

object class.

**Related Topic**

"Using the Schema Displayer" in the chapter "Using DirX Identity Manager" in the *DirX Identity User Interfaces Guide*
Attribute Configuration - Details

## A.6.15. Schema - General Properties

The schema describes the structure of data in a connected directory. A schema consists mainly of a set of object classes and a set of attribute types. Use this tab to assign a name for the schema object.

**Name**

the name of the schema object.

**Version**

the version of the schema object.

**Related Topic**

Schema - Object Classes and Attribute Types

## A.6.16. Specific

### A.6.16.1. JDBC - Configuration

Use this tab to set the following configuration attributes for the JDBC connected directory:

**JDBC Driver**

the class of the JDBC driver that you are using to access a database (the **type** attribute of the Connection element)

**Driver Type**

the driver database customizer (the **driverDBType** attribute of Connection element). This Java class represents the data type capabilities and conversions for the combination of the selected database and the JDBC driver.

**URL**

the URL of the specific database to be accessed (the **url** attribute of Connection element). The URL format is described in the documentation for the JDBC drivers.

### A.6.16.2. GoogleApps - Google API

Use this tab to set the following configuration attributes needed by GoogleApps in order to authenticate the user and authorize access to the Google API:

**Private Key**

the P12 file generated by Google for your service account (Google developer console).

**Service Account Email**

the email generated by Google for your service account (Google developer console).

**Application Name**

the name of the client application accessing the Google service. The name will be used by Google servers to monitor the source of authentication. The name can be anything you chose but the name cannot be blank.

**Domain Name**

must contain the name of your company domain. (If the domain is not configured, it will be determined from the bind profile user ID).

#### A.6.16.3. Office 365 - Graph API

Use this tab to set the following configuration attributes needed by the Office 365 connector to authenticate and authorize access to the Office 365 Graph API:

**Graph API tab:**

**OAuth Path**

The part of the authEndpoint for the OAuth server to get a token. For example: https://login.microsoftonline.com/*TenantID**/oauth2/token*

**API Version**

The version of the Graph RESTful API used by the Office 365 connector. The default value is **v1.0**. Any change might cause incompatibility of the connector with the Graph API. This value is also used for creating the endpoint URL. For example: https://graph.microsoft.com/**v1.0**/groups.

**Bind Profile tab:**

**Application ID**

(client_id) A unique identifier assigned by the Microsoft identity platform that identifies DirX Identity as a client of the Office 365 Graph API.

**Application Secret**

(client_secret) A password or a public/private key pair that your app uses to authenticate with the Microsoft identity platform. (Not needed for native or mobile apps.). Encoded in base64 as generated by PowerShell for client access to the Graph API.

**Tenant**

The tenant value in the path of the request controls who can sign into the application. Possible values are **common** for both Microsoft accounts and work or school accounts, **organizations** for work or school accounts only, **consumers** for Microsoft accounts only, and tenant identifiers such as the tenant ID or domain name. The domain name for the Office 365 tenant as configured in the Office 365 admin center or its tenant ID unique identifier.

#### A.6.16.4. Salesforce - Salesforce

Use this tab to set the following configuration attributes needed by Salesforce Connector to connect to the Salesforce system.

**URL suffixes**

**for OAuth authentication**
: the part of the URL used to authenticate, for example "/services/oauth2/token". Usually there is no need to change this value. This value is combined with the Salesforce's server address, for example "https://login.salesforce.com/services/oauth2/token". After successfully connecting to Salesforce an instance URL is received by the Sales connector, for example "https://na15.salesforce.com" which is used for real service requests (add, delete, modify) later on.

**for Service Requests**
: the part of the URL that is sent for real service requests, for example "/services/data/v30.0". Note that the URL contains the (Rest-)API version that is going to be used by the Salesforce connector, so you should not use a version < "V30.0".

Other parameters:

The Salesforce Connector is running as a remote application and uses OAuth for authentication. Therefore, it must be defined as a new connected app within the Salesforce organization that informs Salesforce of this new authentication entry point. When creating / registering such an app, two new values are created:

**Consumer Key**
: a value used by the consumer to identify itself to Salesforce. Referred to as **client_id** in OAuth 2.0

**Consumer Secret**
: a secret used by the consumer to establish ownership of the consumer key. Referred to as **client_secret** in OAuth 2.0

#### A.6.16.5. OICF - OpenICF Connector Server

Use this tab to set the following configuration attributes to connect and authenticate to the OpenICF connector server:

**OpenICF Server**
: a link to the OpenICF connector server's connected directory, where the name of the OpenICF connector server must be specified in the linked service object.

**OpenICF Server Bind Profile**
: a link to the bind profile of the OpenICF connector server. In the bind profile, specify only the shared secret in the password field. The shared secret key is used to get access to the OpenICF connector server. The same key must be configured at the OpenICF connector server side.

**Response Timeout**

the timeout value in seconds. Set a value that is sufficient for the selected bundle type. The default value is **30**.

**Configuration Attribute Name Mappings**

for OpenICF connector bundles that use non-standard connection property names; for example, "host" instead of "server". Used for building up the connection from the OpenICF connector server to the target machines to be managed. A mapping of these properties can be specified here according to the names the OpenICF connector bundle expects.

# A.7. Jobs

## A.7.1. Authentication

Authentication is necessary to allow the agent to access the related connected directory.Use this tab to set up the authentication data.

**User Name**

the user name for the account that is to run the job.A job can be run under a specific account (the default is the System account).Use this field and the Domain and Password fields to define this account.

**Domain**

the domain where the account must be started.

**Password**

the password of the account.

If the C++-based Server is running under a special account (you cannot use the System account) and an agent shall be run with another account (Authentication tab in the job object), the account of the C++-based sever must have the advanced user rights on the local machine **Act as part of the operating system** and **Replace a process level token**, or the agent cannot run.The steps to set up advanced user rights depend on the type of operating system you are using.

For a Windows server, perform these steps:

- Start **Control Panel** → **Administrative Tools** → **Local Security Policy**
- Select **User Rights Assignment** under **Local Policies**
- Double-click the user rights **Act as part of the operating system** and **Replace a process level token** and add the account as **Local Policy Setting**.
- Reboot the machine.

This account must also be defined as Standard User. Perform these steps:

- Start the Control Panel.
- Start User Accounts.

- Select **Add...**.
- Enter the **User name** and **Domain** of the account.
- Click **Next**.
- Select **Standard User**.
- Click **Finish**.
- Click **OK**.

The account under which the agent can be run after the C++-based Server account has obtained the advanced user rights must satisfy the following requirements:

- The account can be one of the local domain (the domain to which the machine where the agent is started belongs) or it can be an account of any other domain that is trusted by the local domain. If there is only a trust in the direction that the remote domain trusts the local domain but not the other way, the account of the remote domain is not known on the local machine and will produce an error message when used in the Authentication tab.
- In addition to having the right trust relationships between the domains, the account to be used in the authentication tab must also have the appropriate rights on the local machine to be able to run the agent.

**Related Topic**

Job

## A.7.2. Attribute Mapping

Use this tab to edit the items of a Tcl mapping script.A special editor is provided to make the creation or modification of the script as simple as possible.However, basic knowledge about the Tcl language is necessary for all modifications.For usage information on the Tcl Mapping Editor, see the section "Using the Mapping Editor" in the chapter "Using DirX Identity Manager" in the *DirX Identity User Interfaces Guide*.

**Related Topics**

Mapping Item
Mapping Function
Tcl Script
Tcl Content

## A.7.3. Control Scripts

A folder for control scripts that any job configuration object can use.

**Name**

The name of the folder.

**Description**

A description of the folder.

**Related Topics**

(Central) Configuration
Mapping Functions
Mapping Scripts
Other Scripts
Profile Scripts
Tcl Script

## A.7.4. Default Values

This tab allows you to specify default values that NTAgent is to assign to account attributes in the NT system that have no value in the import data file.

See the "Default Values Section" in the chapter "Import Configuration File Format" in the *DirX Identity Connectivity Reference* to learn more about the possible default values.

## A.7.5. Delta Handling

The delta handling information is necessary to control a synchronization of updated entries.Use this tab to set up a delta handling for the job.Available properties are:

**Delta synchronization**

whether (checked) or not (unchecked) a synchronization of updates is required.After you check the box, the other items in the tab can also be edited.

**Delta type**

the calculation base for the update interval.The following types can be used:

**DATE**

the delta is calculated on the basis of a creation or modification date in the connected directory (default value).

**USN**

the delta is calculated using the mechanism of unique serial numbers (for example, as used by iPlanet, Active Directory or OID).

**FILE**

the delta is calculated by some DirX Identity agents (for example, ODBC) that keep the last connected directory state in an external file to compare against the new state.



if you change the **Delta type** property after the job has run an initial delta operation, DirX Identity resets the list of delta runs. You cannot recover this list and you must start with a full synchronization.

**Changes since**

the update interval, which is either:

**(RECENT)**

the synchronization uses the update information of the most recent synchronization procedure.

**(FULL)**

a full data synchronization is performed instead of an update.

If the **Delta type** is **DATE** and at least one delta synchronization has already been performed, you can also choose a date for the interval. The job will synchronize all entries updated between that date and now. If the **Delta type** is **USN**, you can select a USN number for defining the update interval. If the **Delta type** is **FILE**, a file can be selected which contains the update information.

**Elements in list**

the maximum number of delta results that are kept in the data base.

**Clear List**

resets the **Changes since** list. In this case, all delta items are deleted and the next workflow run will perform a full update.

This tab can have additional properties depending on the **Agent** running the job. For the **ODBC** agents, there is the property:

**Data Hash**

a number between 4 and 16 that denotes the octet count used for entry attributes. This hash value is used to find differences between the current value and the value during the synchronization procedure corresponding to the selected delta file.

**Related Topics**

Job

"Microsoft Active Directory Agent" in *DirX Identity Connectivity Reference*
"Novell NDS Agent" in *DirX Identity Connectivity Reference*
"Windows NT Agent" in *DirX Identity Connectivity Reference*
"ODBC Agent" in *DirX Identity Connectivity Reference*
"Hicom DMS Agent" in *DirX Identity Connectivity Reference*
"IBM Notes Agent" in *DirX Identity Connectivity Reference*
"Microsoft Exchange Agent" in *DirX Identity Connectivity Reference*
"SAP ECC Agent" in *DirX Identity Connectivity Reference*
*DirX Identity Application Development Guide* → Understanding the Default Application Workflow Technology → Understanding Tcl-based Workflow → Delta Handling

## A.7.6. Tcl-based Event Manager - Operation



the Tcl-based event manager is only supported for compatibility reasons. Use the Java-based password synchronization instead. It provides more functionality, works on an event-based basis, and uses fewer system resources.

Properties that control the operation of the Tcl-based event manager:

**Notify if not OK**

the notification to make if job problems occur. The item can take one of the following values:

**0** - no notification.

**1** - notification if warnings occurred.

**2** - notification if errors occurred.

**3** - notification if errors or warnings occurred.

**Notifications**

the definitions of how to perform notifications. The entry **NotifyNotOK** keeps the information for the **Notify if not OK** feature. See the section "Understanding Notifications" in the chapter "Managing Provisioning Workflows" for more information.

**Event Messaging Server**

the WebSphere MQ server that is used for event messages. You need to define a dummy C++-based Server object that uses this messaging service if it resides on a machine that does not contain a real C++-based Server representation (installation of this C++-based Server is not necessary!).

**Event Topic**

the events the event manager handles (it listens for this topic at the messaging service). It must be consistent with the definitions in the listeners (for example the Windows Password Listener) and triggers (for example the Web Event Trigger). Use only lower case characters to define the topic. The default topic is **dxm.command.event.pwd.changed.***

**Event Mode**

the types of events on which the event manager listens:

**Events only**

the event manager does not interpret data information contained in the events.

**Events with optional data**

the event manager assumes that the events can contain data (but must not), for example passwords. This data is assumed to be encrypted.

***Must Change Attribute**

the attribute name where to write the "User must change password during next login" from Active Directory. This information can be used at the web application that changes the password, too. It can also be used by workflows to other target systems. If no attribute name is defined, the update operation is not performed and no error or warning is generated. By default, the DirX Identity default applications set the dxmADsResetPassword attribute.

**Maximum Events**

the number of events the event manager waits for before starting the workflows (default: 5).

**Maximum Wait Time**

the amount of time to wait for **Maximum Events** to occur. If the number of Maximum Events does not arrive by the time this limit is reached, the workflows are started anyway (default: 20 seconds). This action occurs only when **Maximum Events** is greater than 1.

**Certificate Retrieval Interval**

the interval at which to check for certificate requests (default 15 minutes) from event listeners.

**Check for System Events**

the interval (in seconds) that advises the event manager to listen for keep alive or shut down requests (delivered as JMS messages via the command queue from the messaging service). The event manager listens to these messages only if the Server Event Support flag of the corresponding workflow object is set. Be sure that this interval is smaller than the Wait Interval of the supervisor.

**Delta Time**

the maximum time differential allowed before automatic correction occurs. DirX Identity can run in a distributed environment, where clocks on the different machines can diverge. Setting the dxmPwdLastChange attribute from a client or the event manager might result in timing differences that can confuse the delta mechanism of the workflows that transport the passwords to the target systems. DirX Identity uses an algorithm that detects these time differences and corrects it automatically. The Delta Time parameter allows you to define the maximum difference allowed before automatic correction takes place. This attribute must be set to a smaller value than the Password Delta Time attribute of all corresponding password synchronization workflows.

**Filter Attribute**

the attribute the event manager uses for filtering events (must be one of the target selected attributes). This field allows for setting up several event managers that distribute load.



There is no built-in consistency check between these event managers. Be sure that the filters are set up correctly in all event managers to handle all events and check that there is no overlap in filters to avoid duplicate processing of events.

**Regular Expression**

the filter expression, in regular expression syntax. See the section **re_syntax** in the Tcl command reference for details about regular expressions.

**Error Wait Time**

the amount of time that the event manager waits before re-starting a workflow that

incurred an error condition (for example, the network was temporarily unavailable). This feature is similar to the retry interval feature of the DirX Identity scheduler.

**Error Repeat Time**

the amount of time the event manager will wait before re-trying to write entries from an in-memory error list to the Identity Store. If events arrive from the Windows Password Listener, the event manager tries to write the attached entries into the Identity Store. If this is not possible (because the entry does not yet exist in the directory), the events are stored in memory in an error list. After Error Repeat Time arrives, the event manager attempts to write the entries from the error list into the Identity Store again.

**Error Repeat Number**

the number of times to retry writing the entries from the error list into the directory. The Error Repeat Time defines the interval between these retries. When the defined number of retries is reached, a notification is sent to the administrator if error notification is enabled. The event is deleted from the error list and from the message queue (which means that the administrator is responsible for solving the problem). If notification is not enabled, the event is removed from the error list but not from the message queue (this means that during the next start of the event manager the event is processed again). To avoid an endless error list, we recommend switching on error notification.

**Check Keep Alive Requester**

the interval at which to check the keep-alive requester. If the event manager listens to system events (see also the Check for System Events flag), it checks that these events arrive regularly. If this is not the case, the event manager assumes a problem at the requester side after the defined time and notifies the administrator about this problem. Note that this feature is only supported when the Server Event Support flag at the corresponding workflow object is set.

**Related Topic**

Event Manager Tracing
Event Manager Workflows

## A.7.7. Tcl-based Event Manager - Tracing



the Tcl-based event manager is only supported for compatibility reasons. Use the Java-based password synchronization instead. It provides more functionality, works on an event-based basis, and uses fewer system resources.

To debug the Tcl-based event manager job, traces are used which are written during operation. Usually, the user can choose a particular trace level to adjust tracing according to the specific needs.

Use this tab so set up trace information of a job. The properties shown in this tab are:

**Debug Trace**

the mode of trace output by the meta controller program. Four different switches can be set in any combination:

**Vars (variable trace)**

traces the variable fields used by the script.

**Cmds (command trace)**

traces all meta controller commands.

**Msgs (message client debug messages)**

debug information from the message client interface.

**Debug (other debug messages)**

other debug messages.

**Trace Level**

the trace level; that is, the granularity of trace output. Supply one of the following values:

**1 - ERROR TRACE** - only failed operations are traced.

**2 - FULL TRACE** - all operations are traced.

**3 - SHORT TRACE** - only operations that access the Identity Store are traced.

**Trace file**

the trace file associated with the job. To display the properties of the selected trace file, click the Properties button on the right.

**Auditing**

whether (checked) or not (unchecked) message auditing is enabled. When checked, the event manager writes log files that report the result of all received messages.

**Report File**

the link to the log file definition. Note that the file name must be in this format: *prefix**.csv
If *prefix* is set to 'audit', the event manager will create two files: audit_ok.csv for all messages where processing succeeded and audit_err.csv for all messages that failed. Note that there might be messages that were first written into the audit_err.csv file. Later on they could be resolved which resulted in an entry in the audit_ok.file.
You can use an tool that is able to read CSV files to examine the resulting audit files (for example Microsoft Excel or Access).

**Statistics**

enables/disables statistics display at the end of the trace file and at the activity and workflow status entries.

**OFF** - no standard statistics output is generated. In this case, the script can optionally output its own statistic information.

**ON** - statistics output is generated (default).

**Related Topic**

Event Manager Operation
Event Manager Workflows

## A.7.8. Tcl-based Event Manager - Workflows



the Tcl-based event manager is only supported for compatibility reasons. Use the Java-based password synchronization instead. It provides more functionality, works on an event-based basis, and uses fewer system resources.

This tab allows you to define the workflows that the event manager is to handle. Use it to assign workflows to this job:

**Workflows**

the list of workflows that the event manager controls. To display the properties of a selected workflow, click the Properties button on the right. Use Add/Remove buttons on the right of the table to add and remove new channels.

**Related Topic**

Event Manager Operation
Event Manager Tracing

## A.7.9. HDMS Parameters

Use this tab to set the properties that control a remote HDMS system.

**HDMS Version**

the HDMS system version. Possible selections are:

HDMS 3.1

HDMS 3.6

HDMS 5.2 (US version)

HiPath 4000 Manager V1.0 *HiPath 4000 Manager V3.1

**Remote Account**

the account on the machine running the remote HDMS database that HDMSAgent is to use to access the HDMS database. Please specify the name of a Reliant UNIX account that has the appropriate permissions for managing the related tables using the XIE import/export program. The default value is “hdmstest”.

Refer to the Hicom DMS documentation for information about how to grant the correct permissions and/or access rights to the HDMS database. The relevant tables are

- PERS, COMPIMP, LOCIMP, BUILDIMP, and ORGIMP for HDMS 3.X
- PERSDAT for HDMS-US 5.2, HiPath 4000 Manager V1.0, and HiPath 4000 Manager V3.1

**Remote XIE program name**

the location of the "remote_hdms" script on the machine that is running the remote HDMS database. HDMSAgent runs the "remote_hdms" script to access the remote HDMS database through the XIE program interface. Please specify the path to the "remote_hdms" script relative to the home directory of the remote account. The default is "bin/remote_hdms".

**Remote XIE data subdirectory**

the subdirectory on the machine running the remote HDMS database that the "remote_hdms" script specified in **Remote XIE program name** is to use for exchanging HDMS XIE request and response files. Please specify the subdirectory path relative to the home directory of the remote account. The default is "req".

For correct setup of the HDMS environment, see the section "HiPath Environment Setup" in the *DirX Identity Application Development Guide*.

## A.7.10. INI Content

Use this tab to edit the content of an INI file.INI files have pure text content and can therefore be edited with any text editor.DirX Identity Manager provides a simple editor for this purpose.The two buttons below the editor enable the user to either export or import INI files.

**Import text...**

click to import a text file which will then replace the current content of the INI file.A file browser dialog is shown to select the file to be imported.

**Export text...**

click to export the current content of the INI file into an external text file.A file browser dialog is shown to select the desired directory and to type in the name of the text file.

**Related Topics**

INI File

## A.7.11. INI File

An INI file is needed to control the operations of a connected directory agent.It contains all necessary parameters needed for a particular synchronization task.The INI file configuration object stores all data needed to create the respective INI file.

Use this tab to set the parameters of the INI file configuration object.The following parameters are available:

**Name**

the name of the INI file.

**Description**

the description of the INI file.

**Version**

the version number of the INI file.

**Content Type**

the content type of the data file. Possible values are:

**UNKNOWN**

an unknown or unspecified content.

**INI**

the data file contains configuration data in an INI file format.

**LDIF**

the content is structured in an LDAP directory interchange format.

**TCL**

the data file contains a Tcl script.

**XML**

the content of the data file is structured in XML format.

**Encoding**

the character encoding of the file.Use any of the valid code sets.See the *DirX Identity Meta Controller Reference* for details.
Note: By default, we use UTF-8 for all meta controller files and ISO-8859-1 for all other files (most agents cannot currently handle UTF-8).

**Anchor**

the text value that helps to select the correct file during reference resolution.See the chapter "Customizing Object References" in the *DirX Identity Customization Guide* for details.

**Related Topics**

Files
Content

## A.7.12. Job

A job is a single executable step in a Tcl-based synchronization procedure used by an activity.The job configuration object holds all necessary properties for the definition of the job, mainly the DirX Identity agent that will carry out this step and the input and output channels that describe the data flow from and to the involved connected directories.A job configuration object describes the configuration information for a particular synchronization job.Use the job configuration object to describe each job that you want to establish for directory synchronization in your Identity environment.

In most cases, the synchronization must be split into two steps:

1. A step that exports the data from source directories and maps it into files
2. A step that maps the content of the files into the data structures of the target directories and imports it into these systems

Since DirX Identity Connectivity considers a file to be a connected directory, every job is treated as a data transfer from source to target connected directories.

The job configuration object can have additional properties depending on the type of agent to which it is connected.

Use this tab to set the general properties of a job. The items shown in this tab are:

**Name**
the name of the job.

**Description**
a description of the job.

**Version**
the version number of the job.

**Agent**
the agent used to run this job. To display the properties of a selected agent, click the properties icon  on the right.

**Operation**
the operation performed by the job (this field is not used by the standard scripts).

**Command Line**
the command line arguments for running the agent associated with the job. Note that this information can contain dynamic references that are resolved before each run of the job.

**OK Status**
the list of all numeric exit code values the job delivers that must be treated as OK. This field allows you to redefine the values defined in the corresponding agent. Several values in this list must be separated by a ';' character.

**Warning Status**
the list of all numeric exit code values the job delivers that must be treated as warnings. This field allows you to redefine the values defined in the corresponding agent. Several values in this list must be separated by a ';' character.

DirX Identity considers all other agent exit codes to represent an erroneous run and stops the workflow's execution.

If the **OK Status** and **Warning Status** properties of a job and the agent have no values, DirX Identity treats each exit code as an error. As a result, you must at least specify one of the agents success exit codes - usually exit code 0 - to make DirX Identity treat it as success.

**Timeout**

the time for the job to run before timing out. The syntax format is *hh:mm:ss*. The default value is 2 hours.

There may be additional properties depending on the **Agent** used for this job.

For the **ODBC** agents, there are the following properties:

**Record Separator**

the string that separates succeeding entries.

**8bit**

only valid for ODBC export. If checked, the agent accepts characters with more than 7 bits without escaping them to hex notation (\x...).

**Related Topics**

Authentication
Delta Handling
Input/Output Channels
Tracing
Notification (Tab)
Specific Attributes

## A.7.13. Jobs

A folder for the job configuration objects in the configuration database.

**Name**

the name of the folder.

**Description**

a description of the folder.

Within a property page, the content of this folder is shown as a pop-up list of a combo box:

**Job** - the job object currently used by the object for which the properties are shown.Use the arrow button to pop up the list of all available job objects in the job folder.Use the properties button to display the properties of the currently selected job object.

**Related Topic**

Job

## A.7.14. Mapping Item

The properties of a mapping item that represents one line of the mapping editor table.We recommend that you do not edit these items directly!Use the mapping editor instead.

**Name**

The name of the mapping item.

**Input**

The input to the mapping function.

**Mapping Function**

The mapping function to use.

**Output**

The output from the mapping function.

**Related Topics**

"Using the Mapping Editor" in the *DirX Identity User Interfaces Guide*

Mapping Scripts
Mapping Functions

## A.7.15. Mapping Script

A mapping script is a special kind of Tcl script that describes the mapping between the source and target selected attributes. It consists of a set of mapping items, each containing one or more source or target attributes and a mapping function which transforms these attributes by appropriate operations into a single target attribute.

The source attributes available for mapping are the selected attributes of the contributing input channel(s) and the selected attributes of the contributing output channel(s). This allows combining source and target attributes with one mapping function. The target attributes into which these source attributes can be converted are the selected attributes of the contributing output channel(s).

DirX Identity provides a mapping editor to create mapping scripts. However, you can also use a standard Tcl editor for the creation or modification of a mapping script.

**Name**

the name of the mapping script.

**Description**

a description of the mapping script.

**Version**

the version number of the mapping script.

**Anchor**

the text value that helps to select the correct script during reference resolution.See Reference Descriptions for details.



Central use of mapping scripts is only possible for Tcl mapping scripts.Table-based mapping scripts must be located directly under the job

object and cannot be centralized.

**Related Topic**

Attribute Mapping

## A.7.16. Mapping Scripts

A folder for mapping script configuration objects that any job can use (when the mapping is not controlled by the mapping editor).



Central use of mapping scripts is only possible for Tcl mapping scripts.Table-based mapping scripts must be located directly under the job object and cannot be centralized.

**Name**

the name of the folder.

**Description**

a description of the folder.

**Related Topics**

(Central) Configuration
Control Scripts
Mapping Functions
Other Scripts
Profile Scripts
Tcl Script

## A.7.17. Notification (Object)

A notification object defines an e-mail notification.Notification configuration objects can either exist at a central location (Configuration folder) or under a job object.You create a notification object with the **Notification** entry in the context menu of either a job object or the notification folder under the Configuration object.The job configuration object provides links to the corresponding notification objects.

A notification object allows you to set up exactly one notification to be used by a job.A notification object consists of the following tabs:

**Notification**

to set the type, the service to be used, all addresses and the attachments to be sent.

**File**

to set the filename of the XML configuration file and some of the other typical parameters for file objects.

**Content**
to handle the XML content of the notification object, including the references.

#### A.7.17.1. Notification Tab

The Notification tab contains the following items:

**Name**
the display name of the notification object

**Description**
a text description of the notification object

**Version**
the version of the notification object

**Type**
the type of notification (only **eMail** is supported)

**Service**
the link to the service object that contains the mail host address.

**From**
the e-mail address in the form *mail=**emailaddress* or the direct link (distinguished name) to the person to be informed when the notification returns because of a bad To or CC address or other problems.

**To**
the e-mail address in the form *mail=**emailaddress* or a link (distinguished name) to the person who is to perform the task.

**CC**
zero or more e-mail addresses in the form *mail=**emailaddress* or links (distinguished names) to the people to be informed about the notification.

**Attachments**
links to files to be sent as attachments with an e-mail notification.

**Anchor**
Sets durable references (for example, use **Data** for data notification; use **NotOK** for notification if not OK).

A prerequisite for all persons whose distinguished names are used in the **From**, **To** and **CC** fields is that an e-mail or another attribute exists which can be used.

#### A.7.17.2. Content Tab

The Content tab allows you to view or edit the XML configuration file, which contains all of the parameters for a notification in an XML structure. Most of these parameters are

referenced to the fields in the Notification tab for easier use. Only the **subject** and the **body** fields must be edited in the XML text directly.



If you want to use person links instead of e-mail addresses, make the following changes directly to the XML configuration file:

- Enter the distinguished name into the fields that are to point to a person, for example, 'cn=Smith John,o=My-Company'. Be sure that the person entry contains an e-mail address.
- In the content tab of the notification object, change the references from "<?Job@dxmNotification-DN@dxmNotifyFrom-DN[5:]/>" to "<?Job@dxmNotification-DN@dxmNotifyFrom-DN@mail/>". If the e-mail address is contained in another attribute, change the last part to this other attribute (for example @my_mail_attribute).



This method works only for user entries that are located in the Connectivity configuration database. It does not work for user entries that are part of another LDAP server.

## A.7.18. Notification (Tab)

The notification tab is part of a job object. Use this tab to set up all notifications for the job. The items shown in this tab are:

**Notifications**

sets the links to all of the notification objects that this job uses. These objects can be local objects under the job objects or central objects in the folder **Configuration** → **Notifications**.

The next three items control whether the add, modify, and delete data change operations are performed automatically (option is not selected) or by a target system administrator who is notified via e-mail.

**Notify to add**

controls all add operations.

**Notify to modify**

controls all modify operations.

**Notify to delete**

controls all delete operations.

When all three items are set, nothing is performed automatically and the target system administrator must perform all operations by hand. A typical scenario is that modifications are done automatically and add and delete operations are done by hand.

**Notify if not OK**

controls notification when a meta controller job encounters an error or warning situation. Possible values are:

**0** - no notification at all

**1** - sends a notification when warnings occur

**2** - sends a notification when errors occur

**3** - sends a notification when warnings or errors occur

## A.7.19. Notifications

A folder for notification objects that any (**metacp**) job can use. Use this folder to define central notification objects.

**Name**

the name of the folder.

**Description**

a description of the folder.

**Related Topics**

Notification (Object)
Notification (Tab)

## A.7.20. Operation

This tab contains most of the attributes that control the operation of a meta controller job that is based on the standard script architecture.

1. Delta Handling

Use these parameters to set up a delta handling for the job.The items shown here are:

**Delta synchronization**

whether (checked) or not (unchecked) a synchronization of updates is required.Once the field is checked, the other items can also be edited.

**Delta type**

the calculation base for the update interval. The following types can be used:

**DATE**

the delta is calculated on the basis of a creation or modification date in the connected directory (default value).

**USN**

the delta is calculated using the mechanism of unique serial numbers (for example, as used by iPlanet, Active Directory or OID).

**FILE**

the delta is calculated by some DirX Identity agents (for example, ODBC) that keep the last connected directory state in an external file to compare against the new state.



if you change the **Delta type** property after the job has run an initial delta operation, DirX Identity resets the list of delta runs. You cannot recover this list, and you must start with a full synchronization.

**Changes since**

the update interval. Select one of the following values:

**(RECENT)**

the synchronization uses the update information of the most recent synchronization procedure.

**(FULL)**

a full data synchronization is performed instead of an update.

If the **Delta type** is **DATE** and at least one delta synchronization has already been performed, you can also choose a date for the interval. The job will synchronize all entries updated between that date and now. If the **Delta type** is **USN**, you can select a USN number for defining the update interval. If the **Delta type** is **FILE**, a file can be selected which contains the update information.

**Elements in list**

the maximum number of delta results that are kept in the database.

**Clear List**

resets the **Changes since** list. All delta items are deleted and the next workflow run will perform a full update.

**Delta Time Attribute**

the attribute to use as an additional filter condition. By default, DirX Identity uses the operational directory attributes **createTimestamp** and **modifyTimestamp** for delta handling when Delta Type is set to Date. You can choose any other time attribute (for example dxmPwdLastChange for password synchronization workflows) to use this attribute as an additional filter condition for delta handling. Note that in a distributed environment time differences can influence the mechanism!

**Password Synchronization**

enables/disables special password synchronization. Setting this flag activates a special mechanism in the standard script that compensates clock differences between the machine where the directory server is running and where clients reside that set the password at user entries in the directory.

**Password Delta Time**

the overlap time used by the special mechanism that is activated by the **Password Synchronization** flag. It must be greater than the Delta Time parameter of all clients

that set the password at user entries.

**Password Sort Key**

the attribute used as the unique ID in the delta files when the special mechanism for password synchronization is used (Password Synchronization flag is on).

## A.7.21. Other Operation Parameters

**Checkpointing Enabled**

enables/disables checkpointing. if this flag is set, the job writes checkpoints regularly. Note: you must also set the Enabled flag at the workflow to enable checkpointing.

**Checkpoint Frequency**

the number of processed records after which a checkpoint is written. After a serious agent problem, this activity can start at the last checkpoint (and not from the beginning again).

**Notify if not OK**

the notification to use if job problems occur. Possible values are:

**0** - no notification.

**1** - notification if warnings occurred.

**2** - notification if errors occurred.

**3** - notification if errors or warnings occurred.

**Notifications**

the definitions of how to perform notifications. The entry **NotifyNotOK** keeps the information for the **Notify if not OK** feature. **NotifyData** stands for a definition for data notification. See the "Understanding Notifications" section in the chapter "Synchronization Procedures" for more information.

**GUID Generation Type**

the type of GUID generation:

**None**

GUID generation is not enabled.

**Local**

generates a GUID composed of the proprietary GUID value from the source connected directory and the GUID prefix.

**Central**

generates a GUID based on the Actual GUID value in the central configuration object.

**User**

- uses user hooks to define your GUID generation algorithm.

**GUID Generation Block Size**

the block of GUIDs that will be handled together if central GUID generation is enabled. This setup minimizes read accesses to the directory server.

**Minimum Source Entries**

the minimum number of entries that must be available for the workflow to run. This variable helps to avoid situations where, during subsequent runs of a full update or export, the number of source entries differs a lot. Use this parameter to specify a minimum amount of entries that must be available or the workflow terminates with error. (Exit code 12 is generated if the number of entries is not sufficient.)

This parameter is not evaluated for import workflows running in **MERGE** mode. It is evaluated for import workflows running in **REPLACE** mode, where it helps to avoid deletion of objects if only a small number of source entries is provided (by mistake).

**Exact Action**

controls the automatic correction features of the algorithm.

**TRUE**

prohibits soft change of action from add to modify or no action when the entry is already deleted (if set to TRUE). Reports an error instead.

**FALSE**

allows soft change of action.

This parameter is only used for import operation.

**Compare with Spaces**

controls whether or not leading, spaces, trailing spaces and any other spaces in the RDNs of a DN should be ignored when comparing DNs.

**TRUE**

take care of spaces when comparing DNs. (This flag should be set if the spaces in the DNs should be taken as they are. RDN values may have been mapped using attributes from a source system where these spaces are very important to be present.)

**FALSE**

ignore spaces when comparing DNs.

This parameter is only used for import operation.

**Init Mode**

the run mode of the job:

**REAL**

the job runs in normal mode and will perform the synchronization step as specified.

**TRIAL**

the job runs just in a trial mode. No update operations will be performed. This mode can be especially used to test the mapping between source and target attributes and to check the shape of the resulting target entries in the trace file.

This parameter is only used for import operation.

**Test Mapping Only**

whether (checked) or not (unchecked) test mapping is used. This field is useful for checking whether the mapping routine works properly. Check the box and then use the Debug Trace switch 2-FILE OUTPUT and the Trace Level 2-FULL TRACE in the Tracing tab to check the mapping output by viewing the trace file in the Monitor View after running the respective workflow.

**Test Max Entries**

the number of source entries to be mapped.This parameter is used with **Test Mapping Only** when this item is enabled.Specify the number of source entries to be mapped and use -1 to map all source entries.

**Related Topics**

Authentication
Input/Output Channels
Tracing
Job
Specific Attributes

## A.7.22. Other Scripts

A folder for Tcl scripts that any job can use (these are Tcl scripts that cannot be classified as control, mapping, or profile Tcl scripts).

**Name**

the name of the folder.

**Description**

a description of the folder.



The load sequence of these scripts is not defined.Thus you cannot define dependencies between these scripts (for example, you cannot overload procedures).

**Related Topics**

(Central) Configuration
Control Scripts
Mapping Functions
Mapping Scripts
Profile Scripts
Tcl Script

## A.7.23. Profile Scripts

A folder for profile scripts that any job can use.

**Name**

the name of the folder.

**Description**

a description of the folder.

**Related Topics**

(Central) Configuration
Control Scripts
Mapping Functions
Mapping Scripts
Other Scripts
Tcl Script

## A.7.24. Tcl Script

The Tcl Script object describes the properties of a Tcl script file, which contains program statements to be run by the DirX Identity controller program **metacp** for synchronization operations.For DirX Identity, there are three main script types:

**Control Scripts**

scripts that contain just definitions and settings of variables later used for the proper synchronization program.Normally all references that relate to fields in various configuration objects are located in this script.

**Mapping Scripts**

scripts that hold a set of items describing the mapping between source and target attributes.See the respective help page for more details.

**Profile Scripts**

scripts that contain the main program.Usually, this is a block of statements to open connections to the source and target connected directory, followed by one or more (nested) loop(s) that are executed for each source directory entry.Such a loop maps the attributes of the entry into target attributes and then tries to join with an existing entry, add a new entry or delete the corresponding target entry depending on which action is requested.

A special editor is provided to create or modify a Tcl script.See the section "Using the Code Editor" in the chapter on DirX Identity Manager in the *DirX Identity User Interfaces Guide* for more details.You may also create or change a Tcl script with a simple text editor, but you will then need to export and import the content.

The attributes shown on this tab are

**Name**

the name of the Tcl script.

**Description**

a description of the Tcl script.

**Version**

the version number of the Tcl script.

**Anchor**

a text value that helps to select the correct script during reference resolution.See Reference Descriptions for details.

**Related Topic**

Tcl Scripts
Tcl Content

## A.7.25. Tcl Scripts

The Tcl scripts used by the job.This tab is only shown when the job is run by the DirX Identity controller program **metacp**.

Use this tab to access the configuration objects of the Tcl scripts used by the job.The properties shown here are:

**Control**

the Tcl script that contains the Tcl control logic.This script is typically the first script that is called from the command line of the job.It calls the profile script, which itself calls other scripts.To display its properties, click the properties icon  on the right.

**Profile**

the Tcl script that contains the Tcl profile logic.To display its properties, click the properties icon on the right.

**Mapping**

the Tcl script that contains the Tcl mapping logic.To display its properties, click the properties icon on the right.

**Miscellaneous**

the Tcl scripts that contain additional Tcl procedures.To display the properties of a selected additional script, click the properties icon on the right.

**Related Topics**

Tcl Scripts
Tcl Content
Job

"Synchronization Profiles and Templates" in *DirX Identity Meta Controller Reference*

## A.7.26. Tracing

You can use traces written during job operation to debug job configurations.In general, you can choose a particular trace level to adjust tracing according to your requirements.

Use this tab so set up trace information for a job.The main field shown in this tab is:

**Trace file**
: the trace file associated with the job.To display the properties of the selected trace file, click the **Properties** button to the right.

This tab can contain other fields depending on the type of the related agent.

The following fields are displayed for the DirX Identity meta controller **metacp**:

**Debug Trace**
: **metacp** trace output mode. Select one of the following values:

  **0 - No Trace**
  : **metacp** will not write any trace output.

  **2 - File Output**
  : **metacp** writes the trace output to the trace file specified in **Trace file** (this level is provided for compatibility reasons).

  **4 - Variable trace to screen**
  : traces the variable fields used by the script.

  **5 - Variable trace to file**
  : traces the variable fields used by the script.

  **8 - Command trace to screen**
  : traces all **metacp** commands.

  **9 - Command trace to file**
  : traces all **metacp** commands.

  **12 - Variable and command trace to screen**
  : combines levels 4 and 8

  **13 - Variable and command trace to screen**
  : combines levels 5 and 9

**Trace Level**
: the granularity of trace output to be generated if **Debug Trace** is set to **2** (**File Output**). Select one of the following values:

  **1 - ERROR TRACE**
  : only failed operations are traced.

**2 - FULL TRACE**

all operations are traced.

**3 - SHORT TRACE**

only operations that access the identity store are traced.

**Max Trace Records**

by default, the meta controller writes only a single trace file (max trace records = 0). Use this counter if you expect to generate very large trace files (more than 2 GB). Set a trace record count, for example 10,000 records, to separate the trace information into multiple trace files. . Don't forget to set up the file name for the trace file with a wildcard (for example, trace*.trc).

**Max Entries**

the maximum number of join hits displayed in the trace file.

**Statistics**

whether or not standard statistics are displayed at the end of the trace file and in the activity and workflow status entries. Select one of the following values:

**OFF**

standard statistics output is not generated. In this case, the script can optionally output its own statistics information.

**ON**

standard statistics output is generated (default).



Trace files from **metacp** are always generated in UTF-8 encoding. As a result, you need a viewer that allows you to view UTF-8 characters. Otherwise the characters are not displayed correctly.

For the ADS and Exchange agents, the only field displayed is:

**Trace**

whether tracing is on (checked) or off (unchecked).

For the IBM Notes agent, the following additional fields are displayed:

**Trace**

whether tracing is on (checked) or off (unchecked).

**Trace Level 1**

whether Trace Level 1 is on (checked) or off (unchecked). Trace level 1 tracing information includes a dump of the configuration file, number of documents, and other program flow variables.

**Trace Level 2**

whether Trace Level 2 is on (checked) or off (unchecked). Trace level 2 provides more detailed information than trace level 1.

**Trace Level 3**

whether Trace Level 3 is on (checked) or off (unchecked). Trace level 3 provides more detailed information than trace level 2.

**Trace Item Types**

whether (checked) or not (unchecked) additional trace information about the types of the Notes database items should be generated; for example:

```
PhoneNumber = TEXT
PasswordChangeTime = TIME
```

For the ODBC agent, the following additional fields are displayed:

**Trace**

whether tracing is on (checked) or off (unchecked).

**Trace Level**

a string that contains a combination of the following items or abbreviations separated by spaces:

**ConnectAttributes (CA)**

include connection "attributes" (fields).

**FailSummary (FS)**

include a summary of failed entries sent to the error file.

**ODBC**

report ODBC versions.

**SQL**

include SQL statements that are to be executed.

**Summary (S)**

include a summary of all entries imported or exported (mark failed entries with a trailing # character.

**Warnings (W)**

include ODBC warnings (ODBC errors are always written to the trace file).

**Columns (Cols)**

include information on columns stored as part of the database schema.

**RefData (Ref)**

include information on the reference data used for delta export.

**Statistics (Stats)**

include statistics about the import operation, such as the number of creates, updates,

and deletes, the number of entries unprocessed because of errors, the total of all entries handled with or without error but not skipped, and skipped entries.

**Max Trace Files**
the number of trace files that the ODBC agents are permitted to create in rotation.

**Max Trace File Size**
the maximum size of a trace file.

**Trace Flow**
the level of tracing information to be written to the trace file. Trace flow is an integer from 0 to 9. The higher the number, the more tracing information is written. Currently, only trace level 1 is implemented; at this level, the ODBC agents give an indication of entrance and exit for main functions.

For the **Report** agent, the following additional fields are displayed:

**Trace**
the trace level. Select one of the following values:

**NO TRACE**
no trace information is written

**STATISTICS ONLY**
only statistic information is written

**FULL TRACE**
all available trace information is written

For **SPML-based** agents (for example **SAP EP UM** agent), the following additional fields are displayed:

**Controller Trace Level**
the trace level for the controller component. Select one of the following values:

**0** - No Trace

**1** - Errors only

**2** - Additional warnings

**5** - Additional informational messages

**6** - Additional debug information

**Connector Trace Level**
the trace level for the connector component. Select one of the following values:

**0** - No Trace

**1** - Errors only

**2** - Additional warnings

**5** - Additional informational messages

**6** - Additional debug information

**Connector Trace Tail**

the number of trace messages to be transferred from the connector to the trace file (default: 20).

**Related Topics**

Job
Files

# A.8. Monitoring

## A.8.1. Activity Status Data

Activity status data are created during and after execution of the associated job for Tcl-based workflows.Use this tab to view the returned status data.

Note that some tabs may be empty if the workflow status entry belongs to a nested workflow.In this case, you must select the link to the child workflow to see the details.

The items shown in this tab are:

**Name**

the name of the entry.

**Activity**

the name of the activity.

**C++-based Server**

the name of the C++-based Server that ran the activity.

**Child Workflow**

the link to the workflow that this activity starts. Click the icon on the right to open this object.

**Start Time**

the activation time for the activity.

**End Time**

the termination time for the activity.

**Status**

the result message returned for the activity (see the topic "Activity Execution Status").

**Remark**

additional activity status information. This field can contain an unlimited number of lines that provide detailed information if the activity did not run correctly. Note that this field

can contain warnings even if the run was successful. If there are warnings, you should determine the reason and fix the problem.

**Exit Code**

the exit code for the activity's execution. This is the exit code returned by the running agent.

DirX Identity captures the exit code of an agent and saves it in the corresponding status entry. By default, DirX Identity considers an activity run to be erroneous when the exit code is not equal to zero.

You can use the **OK Status** and **Warning Status** properties of the agent or job configuration objects to assign specific exit code handling. DirX Identity evaluates all codes that are not specified in these fields as indicating an erroneous run and aborts the workflow.

**Status Path**

the location at which all files defined for this workflow are stored in the file system area. Use the **Copy to Status Area** and **Save Mode** flags in the file item objects to control the status handling.

**Related Topics**

Activity Status - Config
Activity Status Data
Activity Status Trace
Activity Status - Statistics

Tcl-based Activities
Workflows
Workflow Status - Data
Status Data

## A.8.2. Monitor Folder

DirX Identity uses this folder to structure the Monitor View in the same way as it is structured in the Workflows folder of the Expert View.Monitor folders hold other folders or status data for all runs of a particular workflow.

The Monitor Folder tab contains a single item:

**Name**

the name of the monitor folder, usually the name of the workflow.

**Description**

(not used)

Although you can edit this tab, we recommend that you leave the **Name** field as it is, because it is automatically created by the Monitor View.

### A.8.3. Process Table

The process table allows you to monitor running Tcl-based workflows.It contains an entry that can be enabled or disabled for each C++-based Server.When enabled, workflow status entries are displayed as child entries under the corresponding server entry.After the DirX Identity Manager starts, all process table entries are disabled by default.

**Related Topics**

Process Table Server

### A.8.4. Process Table Server

All existing C++-based Servers of a DirX Identity domain are shown as entries under the Process table folder in the Monitor View.

The Monitor tab shows the server state and is refreshed regularly.The other tabs are identical to the tabs of the C++-based server object.

Each C++-based Server entry in the process table has three menu options:

- **Enable Monitoring** - enables monitoring for this server entry.Running workflows will be visible under the corresponding server entry.These entries are marked with a clock icon to indicate that they are still running.This view is automatically refreshed every 30 seconds.Use **Disable Monitoring** to disable it at any time.
- **Refresh** - requests an immediate update of this server entry.
- **Remove Finished Workflows** - enables manual removal of completed workflow status entries.When a workflow monitored in the process table completes, a normal status entry with its icon remains as a child object.You must remove these completed workflow status entries by hand using this option.

Right-clicking on a running workflow aborts the workflow with the option **Abort Workflow** (note: aborting a workflow does not kill running agent processes if the **Abort Execution Allowed** flag is not set in the corresponding agent object before the workflow is started).

**Related Topics**

Process Table

### A.8.5. Workflow Status - Data

A workflow returns status data during and after its execution.You can view this information in the workflow's Workflow Status Data tab in the Monitor View or in the workflow's **Structure** tab.

Note that for Tcl-based workflows, information during execution is only supported if the workflow's **Status Compression Mode** runtime parameter is set to **Detailed**.All other modes deliver status information only after completion of a workflow run.This mode of operation is also valid for the process table: workflow status entries appear only after the run has completed.

Special status entries (with an extension of -E or -E*n*) are used to show conflicts or special situations.See the DirX Identity Manager help topic **Special Error Status Entries** for details.

The following items are shown in the tab:

**Name**
: the name of the file that contains the workflow status data.

**Workflow**
: the name of the workflow that has generated the status data.

**Server Name**
: the name of the C++-based Server or Java-based Server that ran the workflow.

**Parent Activity**
: the link to the activity that started this workflow (for nested Tcl-based workflows). Click the icon on the right to open this object.

**Initiator**
: the initiator of the workflow. For real-time workflows running in the Java-based Server, this field displays either **scheduled** (if activated by an active schedule) or **by event** (if real-time events were sent internally). For workflows running in the C++-based Server, this field displays the name of the relevant schedule or the value defined by the **runwf** tool's initiator switch; if it is empty, the workflow was started interactively from the DirX Identity Manager.

**Start Time**
: the activation time for the workflow.

**End Time**
: the termination time for the workflow.

**Expiration Time**
: the expiration time of the workflow status data. This is the time after which the Status Tracker automatically deletes the data.

**Status**
: the result status returned for the workflow.

**Remark**
: additional workflow status information.This field can contain an unlimited number of lines that provide detailed information if the activity did not run correctly.Note that this field can contain warnings even if the run was successful.If there are warnings, you should determine the reason and fix the problem.

**Related Topics**

Activity
Activity Status Data

Workflows
Workflow Status - Structure

## A.8.6. Workflow Status - Structure

Use this tab to view the control flow of the Tcl-based workflow.The tab displays the sequence of steps (activities), where the colors indicate the following status:

- Yellow - the step has not yet run (this is the default color for all steps at the beginning).
  Corresponding status value: open.notStarted
- Blue - the step is currently running.
  Corresponding status value: open.running
- Green - the step ran successfully.
  Corresponding status value: closed.completed.ok
- Red - the step did not run correctly (Error).
  Corresponding status value: closed.completed.error
- Light red - the step ran correctly but some warnings are reported. You should check what the problems were.
  Corresponding status value: closed.completed.warning
- Gray - the step status cannot be evaluated correctly by DirX Identity (and represents an undefined condition).

Clicking on one of the displayed activities either opens the activity definition (before it runs) or opens the status entry in a new window. This action is especially useful when using the Structure tab in the Global View (you need not switch to the Monitor View).



If the workflow's **Status Compression** M**ode** runtime parameter is set to **Minimized if OK**, no structure information is present.

**Related Topics**

Activity
Activity Status Data
Workflows
Workflow Status - Data

## A.8.7. Workflow Status - Statistics

Use this tab to view statistical information from the complete workflow run.The tab shows summarized information from all related activities (even over nested Tcl-based workflows) as long as the **Disable Statistics** flag is not set.It contains:

**Operation overview**
the type of operation (Add, Modify, Delete) (in rows) and the result (in columns):

**OK**
the number of entries for this operation that were successfully processed.

**Failed**

the number of entries for this operation that failed.

**Ignored**

the number of entries for this operation that were ignored.

The **Total** row and column total the individual results of the operations.The upper left field displays the total number of entries processed.

**Complete Info**

the complete statistical info from the activity run in XML format. This field presents additional information that cannot be displayed in the table format.



Customized scripts for the meta controller may influence the displayed numbers.



If the workflow's **Status Compression mode** runtime parameter is set to **Minimized if OK**, no statistics information is present.

*Example 5. Export activity*

A workflow exports 20 entries from the directory and writes it into a file. In this case, these fields are filled:
Total/Total: 20 (entries read from the source directory)
Add/OK: 20 (correctly processed to the file).

If 3 entries could not be written into the file, the result would be:
Total/Total: 20 (entries read from the source directory)
Add/OK: 17 (correctly processed to the file)
Add/Failed: 3 (not correctly processed to the file)

*Example 6. Import activity*

A workflow imports 50 entries from a file and 20 were added, 15 were modified and 5 were physically deleted.
Total/Total: 50
Add/OK: 20
Modified/OK: 15
Deleted/OK: 5

*Example 7. Transfer activity*

A workflow transfers 30 entries from an LDAP directory to another one and 10 were added (plus 2 upper nodes), 15 were modified and 5 were physically deleted.
Total/Total: 30
Add/OK: 12
Modified/OK: 15

Deleted/OK: 5

**Related Topics**

Activity
Activity Status Data
Workflows
Workflow Status - Structure
Workflow Status - Data

## A.8.8. Activity Status - Config

Use this tab to display the agent configuration files used during operation of a Tcl-based workflow:

**Configuration Files** - the configuration files associated with the activity.To display the contents of configuration file in the list, click it and then click .Use the **Copy to Status Area** flag of a file configuration object to control how it is stored in the status area.

**Related Topics**

Activity Status Data
Activity Status Trace
Workflow Status - Data
Workflow Status - Statistics

## A.8.9. Status Data

This folder is the top-level node for all collected status data in the Monitor View.It contains three types of objects:

- Monitor folders, which keep structured information about running or completed workflows.
- Query folders, which allow for filtering workflow status information.
- The process table, which allows for monitoring running workflows.

1. Activity Status - Input/Output

Use this tab to view the data files created during operation of a Tcl-based workflow:

**Input**

the input files associated with the activity. To display an input file, click it and then click .

**Output**

the output files associated with the activity.To display an output file, click it and then click .

**Related Topics**

Activity Status Data
Activity Status Trace
Workflow Status - Data
Workflow Status - Statistics

## A.8.10. Activity Status - Statistics

This tab displays statistical information from the activity run (currently this feature is only supported by the meta controller in Tcl-based workflows).It contains:

**Operation overview**

displays the type of operation in rows (Add, Modify, Delete) and the result in columns.Result values are:

**OK**

the number of entries for this operation that were successfully processed.

**Failed**

the number of entries for this operation that failed.

**Ignored**

the number of entries for this operation that were ignored.

The **Total** row and column sum the individual results of the operations.



the left upper field displays the total number of file entries processed (this field may be empty for LDAP to LDAP workflows).

**Complete Info**

displays the complete statistical info from the activity run in XML format. This allows defining additional information that cannot be displayed in the Operation Overview table.



Customized scripts for the meta controller may influence the displayed numbers.

*Example 8. Export activity*

A workflow exports 20 entries from the directory and writes them into a file. The following fields are populated:
Total/Total: 20 (entries read from the source directory)
Add/OK: 20 (correctly processed to the file).

If 3 entries cannot be written into the file, the result is:
Total/Total: 20 (entries read from the source directory)
Add/OK: 17 (correctly processed to the file)
Add/Failed: 3 (not correctly processed to the file)

*Example 9. Import activity*

A workflow imports 50 entries from a file and 20 were added, 15 were modified and 5 were physically deleted.
Total/Total: 50
Add/OK: 20
Modified/OK: 15
Deleted/OK: 5

*Example 10. Transfer activity:*

A workflow transfers 30 entries from an LDAP directory to another one and 10 were added (plus 2 upper nodes), 15 were modified and 5 were physically deleted.
Total/Total: 30
Add/OK: 12
Modified/OK: 15
Deleted/OK: 5

**Related Topics**

Activity Status Data
Activity Status Trace
Workflow Status - Data
Workflow Status - Statistics

## A.8.11. Activity Status Trace

Use this tab to view the trace, report and process data files created during a workflow run:

**Trace**

the trace file associated with the activity.To display the trace file contents, click it and then click .

**Report**

the report file associated with the activity.To display the report file contents, click it and then click .

**Process Data**

the process data file associated with the activity.The process data file contains system internal information traced by the agent controller, including the name of the called executable, the command line parameters, and transferred delta information.To display the process data file, click it and then click .

**Related Topics**

Activity Status Data
Activity Status Trace
Workflow Status - Data

# A.9. Password Change

## A.9.1. Password Change Event Manager - Workflow

This workflow reads password change events from an event source, processes them in the LDAP directory and then generates subsequent real-time orders for password change applications to connected directories.

Use this tab to assign properties to the workflow.The items shown in this tab are:

**Name**
the name of the workflow.

**Description**
a description of the workflow.

**Version**
the version number of the workflow.

**Is Active**
whether (checked) or not (unchecked) to enforce running this workflow permanently in the Java-based Server.

**Sending Application**
the application whose events are to be processed. This field allows filtering events based on the event source: the sending application. The asterisk (**) is a wildcard that indicates all senders, *ADS** selects the Windows Password Listener and **Identity** selects the Web Center.

**Type**
the workflow type; in this case, **EventManager**.

**Cluster**
the event source cluster. This field allows filtering events based on the cluster name of the event source. The asterisk (*) is a wildcard that accepts all servers. Note: for Windows sources, this field contains the forest name.

**Domain**
the event source domain. This field allows filtering events based on the domain name of the event source. Enter the appropriate Identity domain; for example, **My Company** for the sample domain.

**Workflow Timeout**
the time after which the workflow stops working if it has not yet completed correctly.

**Edit via content only**

whether (checked) or not (unchecked) you can edit the XML content directly for debugging purposes. When set, the attributes from the various tabs of this object no longer influence the XML content. Do not set this flag for normal operation.

**Related Topics**

Password Change Application - Workflow
Password Change Application - Activity
Password Change Event Manager - Activity
Password Change Event Manager - Workflow
Error Activity

## A.9.2. Password Change Event Manager - Activity

This tab of a password change event manager workflow defines the corresponding activity.

Use this tab to assign properties to the activity.The items shown in this tab are:

**Identity Store**

the Identity Store with which this workflow operates.

**Bind Profile**

the bind profile to bind to the Identity Store.

**Secured via SSL**

whether (checked) or not (unchecked) the workflow uses secure socket layer (SSL) protocol to the Identity Store.

**Write Audit Log**

whether (checked) or not (unchecked) the workflow writes an audit log.

**Number of Retries**

the number of times a password change is automatically repeated after a failure.

**Wait before Retry**

the time to wait between retries.

**Resource Family**

the type of resource the activity needs to run.It runs only on servers that are associated with the same resource family or families.

**Related Topics**

Password Change Application - Workflow
Password Change Application - Activity
Password Change Event Manager - Activity
Password Change Event Manager - Workflow
Error Activity

### A.9.3. Password Change Application - Workflow

This workflow is triggered by password change orders created by the password change event manager.It processes them correctly to the corresponding connected directory.

Use this tab to assign properties to the workflow.The items shown in this tab are:

**Name**

the name of the workflow.

**Description**

a description of the workflow.

**Version**

the version number of the workflow.

**Is Active**

whether (checked) or not (unchecked) to enforce running this workflow permanently in the Java-based Server.

**Type**

the type of the workflow. Identical to the type of target system this workflow handles.

**Cluster**

the cluster name of the target system. This value must be exactly the same as the value in the **Cluster** field of the associated target system entry in the DirX Identity domain (Advanced tab). Use the asterisk (*) as a wildcard to accept any server name. Note: for Windows target systems, this field contains the forest name.

**Domain**

the domain name of the target system. This value must be exactly the same as the value in the **Domain** field of the associated target system entry in the DirX Identity domain (Advanced tab). Use the asterisk (*) as a wildcard to accept any domain name.

**Workflow Timeout**

the time after which the workflow stops working if it has not yet completed correctly.

**Edit via content only**

whether (checked) or not (unchecked) you can edit the XML content directly for debugging purposes.When set, the attributes from the various tabs of this object no longer influence the XML content.Do not set this flag for normal operation.

**Related Topics**

Password Change Application - Workflow
Password Change Application - Activity
Password Change Event Manager - Activity
Password Change Event Manager - Workflow
Error Activity

### A.9.4. Password Change Application - Activity

This tab of a password change application defines the corresponding activity.

Use this tab to assign properties to the activity.The items shown in this tab are

for Java Connectors:

**Connected Directory**
: the connected directory where the password change is to be performed.

**Bind Profile**
: the bind profile to use to connect to the connected directory.

**SSL**
: whether (checked) or not (unchecked) secure socket layer (SSL) protocol is used to the connected directories.

**for C Connectors:**

**Related IdS-C Server**
: the relevant C++-based Server. This activity communicates via SOAP with a C++-based Server. The connector running in this server performs the password change action at the connected directory (the API of this connected directory is only accessible via a C or C++ interface).

**Connector**
: the relevant connector running in the IdS-C server.

**SSL for SOAP**
: whether (checked) or not (unchecked) SSL protocol is used for the SOAP connection.

Common Attributes:

**Write Audit Log**
: whether (checked) or not (unchecked) the activity writes an audit log.

**Number of Retries**
: the number of times the activity repeats the operation if a password change fails.

**Wait before Retry**
: the time between retries.

**Password Reset Attribute**
: the attribute used in some of the connected directories to indicate a password reset (requires the user to change the password during next login at this connected directory).

**Mapping Script**
: the JavaScript that defines the mapping between the attributes in the Identity Store and in the connected directory.

**Resource Family**

the type of resource this activity needs to run.It runs only on servers that are associated with the same resource family or families.

**Related Topics**

Password Change Application - Workflow
Password Change Application - Activity
Password Change Event Manager - Activity
Password Change Event Manager - Workflow
Error Activity

## A.9.5. Error Activity

This activity only receives requests that have permanently failed.It is optional.When it is not enabled, the failed requests are placed in the Java-based Server's dead letter queue.

The activity sends notifications to the affected users.It builds the notification and optional attachments by reading attributes from the request such as user's e-mail or name.

Use this tab to assign properties to a workflow.The items shown in this tab are:

**Enabled**

whether (checked) or not (unchecked) an error activity is included into the workflow that sends notifications.

**Error Script**

the JavaScript that generates the attributes of the SPML AddRequest passed to the mail connector.The mail connector uses these attributes to build the e-mail and the attachments.

**Resource Family**

the type of resource this activity needs to run. It runs only on servers that are associated with the same resource family or families.

**Related Topics**

Password Change Application - Workflow
Password Change Application - Activity
Password Change Event Manager - Activity
Password Change Event Manager - Workflow
Error Activity

## A.9.6. Error Notification

This activity only receives requests that have permanently failed. It is optional. When it is not enabled, the failed requests are placed in the Java-based Server's dead letter queue.

The activity sends notifications to the affected users. It builds the notification and optional attachments by reading attributes from the request such as the user's e-mail or name.

Use this tab to assign properties to a workflow. The items shown in this tab are:

**SMTP Server**
the mail server to use for sending notifications.

**Bind Profile**
the bind profile for this mail server (optional).

**From**
the "From" field of the mail to be sent.

**To**
the "To" field of the mail to be sent.

**Subject**
the "Subject" field of the mail to be sent.

**Body**
the "Body" field of the mail to be sent. You can use these variables:

**${response(errormessage)}**
the error message.

**${IDATTR(id)}**
the identification of the user (either the DN or the username).

**Related Topics**

Password Change Application - Workflow
Password Change Application - Activity
Password Change Event Manager - Activity
Password Change Event Manager - Workflow
Error Activity

# A.10. Java-based Workflows

## A.10.1. Combined Java-based Workflow

A "combined" Java-based workflow is a workflow that is composed of multiple individual Java-based workflows. You can build a combined workflow to include existing Java-based workflows and define the sequence in which they should run. You can also specify how each workflow is to operate when a WARNING status is returned by a preceding workflow or activity in the sequence. Note that you cannot use entry change workflows, cluster workflows or other combined workflows in a combined Java-based workflow definition.

Use the Workflow Sequence tab of a combined Java-based workflow configuration object to create and manage a combined Java-based workflow. This tab consists of a table on the left and a toolbar for operating on the table on the right.

#### A.10.1.1. The Workflow Table

The workflow table consists of two columns:

- Workflows - the workflows that comprise the combined workflow. Each row in the table specifies one Java-based workflow. The order of the workflows in the table determines the sequence in which each workflow is run. Click in a row to select it. Click  in a workflow row to view the workflow's properties. Click  to browse to and select a workflow.
- StopOnWarning - whether (**true**) or not (**false**) this workflow or activity is not started if its predecessor (the workflow in the row above it) finishes with a WARNING status. A value of **true** means that the entire combined workflow stops on a WARNING returned from the preceding workflow or activity in the sequence. A value of **false** (the default) means that this workflow or activity starts after its predecessor finishes regardless of whether it finished with WARNING or SUCCESS. Click in the row to toggle between **true**/**false** value selection.

#### A.10.1.2. The Toolbar

The actions for operating on a selected row in the workflow table are:

 **Add new workflow** - inserts a new row below the current selection. In the Workflow column of the new row, click  to browse to the workflow you want to add.

 Delete - deletes the selected row.

 Duplicate - duplicates the first selected row.

 Move up - moves the selected row up.

 Move Down - moves the selected row down.

 Copy - copies the selected row to the clipboard.

 Paste - pastes the selected row from the clipboard.

**Related Topics**

Real-time Activity
Content Tabs
Java-based Workflow

### A.10.2. Java-based Workflow

A Java-based workflow consists of one or more activities that carry out part of a real-time or scheduled data synchronization operation.A workflow configuration object describes the configuration information for a particular workflow including the configuration information for all included objects like activities.Use the workflow configuration object to define Java-based workflows.

You can find all relevant activities of **Provisioning Synchronization workflows** as sub-

objects of the workflow configuration object.

**Password Synchronization workflow** objects written in JavaScript technology also contain the information about the included activities, such as:

- The productive **setPasswordActivity**, which implements the task that the workflow performs.It contains information about the connected directory to work with, operation information and the resource family to which it belongs.
- The **Error Activity**, which defines actions that occur when the productive activity fails.This information includes operation information and the notification definition.



If you have changed a workflow configuration, inform the server to reload it! Select the workflow configuration object and choose "Load IdS-J Configuration" from the context menu.

Use this tab to enter the properties of a workflow object (note that not all items are present for a specific workflow). The items shown in this tab are:

**General**

**Name**

the name of the object. The name is limited to 45 characters. This limit avoids a monitor entry cn with more than 64 characters. The cn for the monitor entry is: *name_of_workflow 16_char_timestamp 2_optional_characters*-**C**

**Description**

a description of the object.

**Workflow Type**

the type of the workflow (only visible if design mode is enabled). This field is handled as a comment (there is currently no functionality associated with it).

**Is Active**

whether (checked) or not (unchecked) to enforce running this workflow permanently in the Java-based Server.

**Associated TS**

the DN of the "real" target system for retrieving user data. This field is only required for connected systems such as portals that implement single sign-on (SSO) for a number of other applications. If synchronization to the portal system requires user data stored in the entries of the "real" system, the DN of the corresponding target system needs to be entered here.

**Is applicable for**

the application of this workflow:

**Topic Prefix (read-only)**

the topic prefix for the event that is used to trigger this Java-based workflow. To determine whether the prefix is appropriate, compare it with the topics listed in the

Topic entries in folder Configuration → Topics.

**Type**

the type of workflow. This value is identical to the type of target system this workflow handles.

**Cluster**

the cluster name of the target system. This value MUST be exactly the same as the Cluster field of the associated target system entry in the DirX Identity domain (Advanced tab). Wildcard "*" accepts any server name. This field must be empty if the workflow runs only in scheduled mode.
Note: in some target systems, this field is named differently (for example 'Forest Name' for the Windows target systems).

**Domain**

the domain name of the target system. This value MUST be exactly the same as the Domain field of the associated target system entry in the DirX Identity domain (Advanced tab). Wildcard "*" accepts any domain name. This field must be empty if the workflow runs only in scheduled mode.

**Timeout**

the workflow timeout. The workflow engine moves the workflow to the dead letter queue if the timeout is reached.
Note: be sure to set all activity timeouts correctly. Activity and workflow timeouts are completely independent. The first timeout that is reached forces the server to abort the workflow.

**Endpoints**

the connected directory types supported by this workflow. This information allows DirX Identity to determine which workflows fit between the source and target connected directories that are connected with a workflow line in the Global View. Set the value to the correct connected directory types with a "-" in between, for example "LDAP-ADS". If set correctly, this workflow appears in the dialog of the **Assign** or **New** (copy) action at a workflow line.
Note: If your workflow does not appear at the desired line, check the types of the connected directories and simply change the Endpoint field accordingly. Now you can assign or copy the workflow. Do not forget to change the Endpoints field to the previous value.
For example, suppose you want to assign a consistency workflow that handles the assocAccount2User rule to a workflow line that connects the Identity Store and an ADS. The consistency workflow is defined as "LDAP-LDAP". To add it to the workflow line, change the value to "LDAP-ADS", and then you can assign the workflow to the line. After this step, reset the Endpoints field to "LDAP-LDAP". This method allows you to build more intuitive user interfaces.

**Wizard**

the wizard used for configuring the most important parts of this real-time workflow.

**Related Topics**

Real-time Activity
Real-time Channel
Content Tabs
Real-time Filter
Real-time Java Mapping
Real-time Port

See also "Managing the Java-based Server".

## A.10.3. Real-time Activity

A real-time activity is a single step in a real-time workflow.The activity configuration object maintains all data needed for the activity's execution within the running workflow.The configuration data associated with an activity includes:

- The activity's name, description and version number
- Information about type, implementation, error handling and auditing

Activity objects contain port objects that define how to access the connected directories and their information.

Use this tab to enter the properties of an activity object.The items shown in this tab are:

### A.10.3.1. General

**Name**
the name of the object.

**Description**
a description of the object.

**Version**
the version number of the object.

**Job Type**
the naming attribute from the component description (only visible if Design Mode  is enabled). The values are:
errorActivity - error job for an activity
provisioningActivity - provisioning job for an activity
pwdHandlingActivity - job for password expiration handling
resolutionActivity - job in an event maintenance workflow
transportActivity - job of a collection export / import activity

**Resource Family**
the type of resource this activity needs to run. It runs only on servers that are associated with the same resource family or families.

**Timeout**
the workflow timeout. The workflow engine moves the workflow to the dead letter

queue if the timeout is reached.



Be sure to set all activity and workflow timeouts correctly. Activity and workflow timeouts are completely independent. The first timeout that is reached forces the server to abort the workflow. Activity timeouts can accumulate regarding the **Retry limit** and **Wait before retry** settings.

**Retry Limit**

the number of retries if the activity ran on a recoverable error.

**Wait before retry**

the time between retries.

#### A.10.3.2. Controller

**Join Engine Type**

the type of join engine used in this job. Read the *DirX Identity Application Development Guide* for an explanation of the available controller types.

**Class Name (Controller)**

the class name of the controller to be used.

**Write Audit Log**

whether (checked) or not (unchecked) auditing is enabled for this activity.

**Userhook Class Name**

the class name of the user hook to be used.The user hook class must implement the interface **com.siemens.dxm.join.api.IGlobalUserHook** and must be deployed to **${DIRXIDENTITY_INST_PATH}/ids-j-***domain***-S***n***/confdb/common/lib**.

**Error Script (optional)**

the link to the error script to be used.

**Implementation Language (optional)**

the implementation language of the error script.

**Send E-mail (optional)**

whether (checked) or not (unchecked) a notification is sent when a user changes his own password. This field is used in the User Password Event Manager workflow. If set and the user changes his own password, a notification is sent (note that if an administrator resets a user's password, a notification is always sent).

**Filter for Accounts** (optional):

**Search Base**

the node at which to start the search.

**Scope**

the scope of the search.

**Filter**
the LDAP filter to use to restrict the search.

Other optional fields:

**Days before Expiration (optional)**
the number of days before the password expires.

**Keep Password History at the Account (optional)**
a flag whether (checked) or not (unchecked) a password history should be kept at the account (the default is false unless it is a privileged account).

**Number of Notifications (optional)**
the number of notifications that should be sent.



Some of these properties are only visible for specific controller implementations.

#### A.10.3.3. Notification (optional)

**From**
the e-mail address of the sender.

**To**
the e-mail address of the receiver.

**Subject**
the title (subject) of the e-mail.

**Body**
the body of the e-mail.

**Related Topics**

Real-time Channel
Content Tabs
Real-time Filter
Real-time Java Mapping
Real-time Port
Java-based Workflow

See also "Managing the Java-based Server".

### A.10.4. Real-time Port

A port configuration object defines the access to a connected directory that is used by the corresponding activity.

The configuration data associated with a port includes:

- The port’s name, description and version number
- Information about type, authentication data as well as links to the related channels.

Port objects contain references to channel objects that define more information about joining and mapping.

A real-time port defines the connection to a connected directory for several object types (for example, accounts, groups, memberships). The port configuration object defines bind parameters and keeps channel references. Object type-specific information (for example, the mapping) is defined in channels. A port can reference several channels.

Use this tab to enter the properties of a port object. The items shown in this tab are:

**General**

**Name**
the name of the object.

**Description**
a description of the object.

**Version**
the version number of the object.

**Port Type**
the type of port. Possible values are:
errorPort - port for error notification
notificationPort - port for normal notification (for example, to notify adds and deletes to an administrator)
provisioningPort - synchronization ports to a target system or the Identity Store

**Target System**

**Port Name**
the name of the port. Only a fixed list of names is supported:
TS - the port that connects to the connected system.
IdentityDomain - the port that connects to the Identity Store.
event - the port that sends change events to the DirX Identity message broker.
notify - the port that sends e-mail notifications.
(Message - the port that sends e-mail notifications (deprecated - used only in older workflows)).

**Connector**
the type of connector for this port.

**Channel Parent**
the channel parent folder (a subfolder under the channel folder of a connected directory). It contains all related channels for this port.

**Attributes to Decrypt**

the attributes to be decrypted.

**Connector Class Name**

this field is automatically set when the connector type is selected.

**Connected Directory / Mail Server**

the connected directory this port allows you to access.

**Bind Profile**

the bind parameters to access the connected directory.

**SSL**

whether (checked) or not (unchecked) to use SSL. This field is valid only for certain workflow types like those of type ADS. The SSL flag is set here at the port because it is workflow specific. Validation workflows must not be run with SSL, but synchronization workflows, which need SSL to be able to set passwords in the connected system, must be run with SSL.

**Message Server**

the messaging service to which events should be sent. JMS connectors - such as the one sending change events - need to know to which messaging service they should send events. This is typically the DirX Identity message broker.

**Publisher ID**

the publisher ID. JMS connectors typically need a publisher ID that distinguishes them from other JMS publishers.

**OpenICF Connector Bundle (only for workflows of type OpenICF)**

**Bundle Specification**

**Bundle Name**

the name of the OpenICF Connector bundle running inside the OpenICF Connector Server.

**Bundle Version**

the version of the OpenICF Connector bundle.

**Class Name**

the class name of the OpenICF Connector bundle to be called by the OpenICF Connector Server.

**Request Workflows (optional)**

**URL Path**

this field is pre-configured to the default value **workflowService/services/WorkflowService**. Do not change this setting.

**Socket Timeout (optional)**

the timeout (in seconds) if necessary.

**Domain (optional)**

the domain for which this Java server is responsible. this field helps to verify whether the HTTP request to the workflow service goes to the correct Java server that is responsible for this domain. If this field is empty, no check is performed. We recommend to set the domain here.

**Primary Request Workflow (optional)**

the request workflow to be used. It will be used for account objects if configured. If no Secondary Request Workflow is set, it will be used for group objects, too. If the field is empty

**Secondary Request Workflow**

the request workflow to be used for group objects.



if both primary and secondary workflow fields are empty, the workflow service tries to find a suitable request workflow definition according to the **When Applicable** settings. It is recommended to configure at least a primary workflow. Be aware that usage of a single request workflow definition requires an implementation that is able to handle both object types.

**Notification (optional)**

For Java-based real-time workflows (other than Set Password workflows), notifications are defined as follows:

**Subject**

the title (subject) of the e-mail.

**From**

the e-mail address of the sender.

**To**

the e-mail address of the receiver.

**CC**

the e-mail address of the copy recipient.

**BCC**

the e-mail address of the blind copy recipient.

**Subject**

the title (subject) of the e-mail.

**Body**

the body of the e-mail.

**Batch Size**

number of e-mails that are collected before e-mails are sent.



- When using the User Password Expiration Notification workflow, the following expressions can be used:

  ${IDATTR(cn)} - the user's common name

  ${IDATTR(daysToExpire)} - the number of days before the user's password expires

  ${IDATTR(expirationDate)} - the expiration date of the user's password

  ${IDATTR(givenName)} - the user's given name

  ${IDATTR(mail)} - the user's mail address

  ${IDATTR(sn)} - the user's surname

- When using the User Password Event Manager workflow, the following expressions can be used:

  ${IDATTR(cn)} - the user's common name

  ${IDATTR(givenName)} - the user's given name

  ${IDATTR(id)} - the user's identifier (for example, DN for LDAP sync.)

  ${IDATTR(mail)} - the user's mail address

  ${IDATTR(password)} - the user's encrypted password

  ${IDATTR(passwordexpired)} - password reset flag

  ${IDATTR(sn)} - the user's surname

- When using expressions for the recipients, make sure that the expressions will not resolve to "" (for example, the attribute "mail" should be present); in that case no e-mails are sent if none of the recipient fields is present.
- In error notifications, the following expression can be used in the **Body** field:

  ${response(errormessage)} - the error message

**Notification (optional)**

For Java-based Set Password workflows (when the user has changed his own password using self service), the notifications are defined as follows: (the recipients are defined in the Recipients tab):

**Subject**

the title (subject) of the e-mail.

**Body**

the body of the e-mail.

**Batch Size**

the number of e-mails that are collected before e-mails are sent.



In the **Subject** and **Body** fields, the following expressions can be used:

${IDATTR(cn)} - the user's common name

${IDATTR(givenName)} - the user's given name

${IDATTR(id)} - the user's identifier (for example, DN for LDAP sync.)

${IDATTR(mail)} - the user's mail address

${IDATTR(_originatinguser)} - the DN of the user initiating the password update

${IDATTR(_originatingusercn)} - the originating user's common name

${IDATTR(_originatingusergivenname)} - the originating user's given name

${IDATTR(_originatingusermail)} - the originating user's mail address

${IDATTR(_originatingusersn)} - the originating user's surname

${IDATTR(password)} - the user's encrypted password

${IDATTR(passwordexpired)} - password reset flag

${IDATTR(sn)} - the user's surname

${IDATTR(_tscn)} - the common name of target system where the user's password has been updated

${IDATTR(_tsid)} - the DN of target system where the user's password has been updated

${IDATTR(_usercn)} - the user's common name

${IDATTR(_usergivenname)} - the user's given name

${IDATTR(_usermail)} - the user's mail address

${IDATTR(_usersn)} - the user's surname

In error notifications, the following expression can be used in the Body field:

${response(errormessage)} - the error message

**Notification on Reset (optional)**

For Java-based Set Password workflows (when an administrator has reset a user's password), the notifications are defined the following way: (the recipients are defined in the Recipients tab)

**Subject**

the title (subject) of the e-mail.

**Body**

the body of the e-mail.

**Batch Size**

the number of e-mails that are collected before e-mails are sent.



The same expressions as listed above (when a user changes his own password) can be used.

**Recipients (optional)**

For Java-based Set Password workflows, the recipients used in the notification e-mails are defined in the following way:

**From**

the e-mail address of the sender.

**To**

the e-mail address of the receiver.

**CC**

the e-mail address of the copy recipient.

**BCC**

the e-mail address of the blind copy recipient.



- The same expressions as listed above can be used; the only expression for recipients that makes sense is "$\{IDATTR(mail)}". Of course one could define expressions like "$\{IDATTR(givenName)}.$\{IDATTR(sn)}@My-Company.com" but due to hard coded domain suffix this is not so useful.
- When using expressions for recipients, make sure that the expressions will not resolve to "" (for example, the attribute "mail" should be present); so that e-mails are not sent if no recipient fields are present.



Some of these fields are optional for some port types.

**Related Topics**

Real-time Activity
Real-time Channel
Content Tabs
Real-time Filter
Real-time Java Mapping
Java-based Workflow

See also "Managing the Java-based Server".

## A.10.5. Real-time Filter

A real-time connector filter can be configured to be injected into a filter pipe between the join engine and the connector.DirX Identity supports some specialized filters and a generic filter to be used for custom implementations.

Specialized filters are:

**CryptFilter**
: encrypts / decrypts attributes in requests and responses.

**JDBCFilter**
: (for JDBC-type connected systems) - transforms multi-value membership attributes in Identity to multiple records in the JDBC membership table.See the JDBC workflow for more details.

Use this tab to enter the properties of a filter object.The items shown in this tab are:

**General**

**Name**
: the display name of the filter.

**Description**
: the description of the filter.

**Sequence number**
: the location of this filter, if more than one filter is configured. This value must be a unique positive integer. The first filter is the first to receive the request from the join engine.

**Filter**

**Name**
: the name of the filter. Stored only in XML content (no further use).

**Class Name**
: the fully qualified class name of the filter implementation.

**Use PSE**
: whether (checked) or not (unchecked) the filter is allowed to read the Crypto PSE for decryption.

**Properties**
: the list of custom attributes to be entered with their name and String values.

The specialized filters require additional attributes:

**Attributes for the JDBCFilter:**

**Member Table**

the database table in which to store the account-group memberships; that is, the values of the member attribute.

**Member Source Attribute**

the column name of the database table that identifies the entry holding the list of account-group memberships. Typically this is the account.

**Member Attribute**

the column name of the database table that identifies the member. Typically this is the group.

**Match Type**

the column name of the database table that identifies the entry holding the multi-value attribute. Typically this is the account.

**Multivalue Attributes**

the column name of the database table that holds the values of the multi-value attributes. They should be accumulated and returned as one multi-value attribute in the SPML search response entry.

**Attributes for the CryptFilter:**

**Decrypt Request**

whether (checked) or not (unchecked) the filter evaluates each request for attribute values to decrypt.

**Decrypt Attributes**

the (comma separated) list of attributes whose values are decrypted. Typically this field contains an expression that references the specific attribute "decrypt.attribute" of the port entry: $\{DN4ID(port)@dxmSpecificAttributes(decrypt.attribute)}. The values are entered at the port.

**Encrypt Attribute**

the attribute to decrypt/encrypt.

**Related Topics**

Real-time Activity
Real-time Channel
Content Tabs
Real-time Java Mapping
Java-based Workflow

## A.10.6. Real-time Channel

A real-time channel configuration object is a sub-object of a connected directory that defines the attributes, the attribute mapping, the export and the join filter for a certain type of objects (for example, users or groups) stored in this connected directory.

Java mapping objects can be sub-objects of channel objects if the Java Source mapping type is used.

Note that for secondary channels such as the password or member channel, you don't need to specify export criteria, join conditions or import options.They are taken from the corresponding primary channel, typically the channel for accounts (for the password channel you define the primary channel with the Password Primary Channel link, for the member channel you define the secondary channel with the Member Channel link form the primary channel).In the secondary channels, you only need to define the mappings for the appropriate attribute(s); that is, the member attributes for group memberships or the password attribute for the password channel.

Use this tab to enter the properties of a real-time channel.The items shown in this tab are:

#### A.10.6.1. General

**Name**
the name of the object.

**Description**
a description of the object.

**Version**
the version number of the object.

**Export Sequence Number**
the processing sequence of the channels during runtime. Use 0 for a member channel.

**Connected Directory**
the link to the relevant connected directory.

**Corresponding Channel**
the corresponding channel in the other direction.

**Member Channel**
the relevant member channel. Set this value only for those channels (the primary channels) that hold the members. In LDAP based systems, these are the groups.

**Object Description Name**
the object description name of the provisioned object to use to find the associated audit policy to determine whether audit messages must be written. It is also used for finding the associated change event policy if change events need to be sent.

**Password Primary Channel**
the primary channel that keeps the export criteria, join conditions or import options which are reused by the set password channel.Entering a value this field is only necessary for a set password channel.

**Userhook Class Name**

the class name of the user hook definition.The user hook class must implement the interface **com.siemens.dxm.join.api.IUserHook** or **com.siemens.dxm.join.api.IUserHookExt** and must be deployed to **${DIRXIDENTITY_INST_PATH}/ids-j--***domain***-S***n***/confdb/common/lib**.

#### A.10.6.2. Import

**Include ID in Add Request**

whether (checked) or not (unchecked) to include the ID in the add request. Some target systems require the ID as part of the add request.

**Create Despite Multiple Joined Entries**

whether (checked) or not (unchecked) to create the entry even when multiple joined entries are found during the join operation. If you deselect this option, the workflow reports an error and does not create the entry.

**Notify on Add**

whether (checked) or not (unchecked) to send a notification message in addition to the add operation. See the notify port for configuration of the message.

**Notify on Delete**

whether (checked) or not (unchecked) to send a notification message in addition to the delete operation. See the notify port for configuration of the message.

**Notify on Modify**

whether (checked) or not (unchecked) to send a notification message in addition to the modify operation. See the notify port for configuration of the message.



The Set Password workflows are the only workflows that evaluate this flag. The other real-time workflows are not normally interested in each modification.

If an administrator resets a user's password, the notification is always sent. If the user changes his own password, the notification will only be sent if **Notify on Modify** is set.

#### A.10.6.3. Export

This section describes the search parameters needed to export the objects for this channel. When the workflow runs triggered from the scheduler, it uses them to search all relevant objects. When triggered from a single event, the join engine uses them to associate the matching channel for the changed object.

**Scope**

the scope of the search.

**Search Base Type**

the identifier type for the search base definition below according to the SPML definition

(OASIS). Valid types are:

**DN**
the identifier is of type distinguished name.

**EMailAddress**
the identifier is of type e-mail address.

**GUID**
the identifier is of type global unique the identifier.

**GenericString**
the identifier is of type generic string.

**LibertyUniqueID**
the identifier is of type unique ID according to the Liberty definition.

**OID**
the identifier is of type object the identifier.

**PassportUniqueID**
the identifier is of type unique ID according to the Passport definition.

**SAMLSubject**
the identifier is of type subject according to the SAML definition.

**URN**
the identifier is of type unique resource notation.

**UserIDAndOrDomainName**
the identifier is either a user ID or a domain name.

**Search base**
the identifier of the base node for the search. Its definition is very similar to simple mapping expressions. You can define one or more expressions, where each is a concatenation of strings:

*expr* [**+** *expr*]

where *expr* stands either for a variable or a string constant.
Variable can be:

**${env.***name***}** - variables taken from the environment, where *name* defines the attribute name.

A string constant is enclosed in double quotes (for example, "*constant*").

Examples:

```
${env.user_base}
"ou=sales" + $\{env.user_base}
"cn=Accounts,cn=Extranet Portal,cn=TargetSystems,cn=My-Company"
```

**Filter**

the filter of the search. Note that only attributes used or mentioned in the mapping definition on the right side of the mapping table (**Map to**) should be used in a filter expression. (See "Java Mapping Editor" for details.) Otherwise, filtered attributes will not be available during the filter evaluation and the filter may deliver incorrect results. If you do not intend to use the filtered attribute in the workflow at all, use the following special mapping. If the attribute **attributeName** should be used in a filter expression but does not occur on the right side of any of the existing lines in the mapping editor, specify a new **Direct** mapping with **null** value mapped to **attributeName**. Then set the flags **retrievable** and **readOnly** for it. Such an attribute can be then correctly evaluated in a filter expression.

The value of a filter conditions can be either a plain string (no **$\{** and **}** contained) or expression. The expression is a concatenation of sub-expressions:

*expr* [**+** *expr*]

where *expr* stands either for a variable or a string constant.
Variable can be:

**${env.***name***}** - variables taken from the environment, where *name* defines the attribute name.

A string constant is enclosed in quotes (for example, "*constant*"). In the GUI editor, the quotes must be doubled due to LDAP escaping (""*constant*"").

Examples:

```
${env.my_value}
"ou=sales" + ${env.my_sales}
```

**Filter (Office 365 Connector)**

This filter can only be used in a limited way.
The filter is only available for **accounts**, **groups**, and **roles** channels.
Not all AD objects own the attributes defined in **Attr-Conf-File**.

***Operators***

The following filter operators are not supported: **“not”**, **“is present”**, **“contains”**, **“ends with”**. The roles channel filter only supports the operator **“equals”**.

***Attributes***

Not every Azure AD Object (user, group) property supports filter query. Check the Microsoft documentation for the resource to see which property is filterable. Only the

properties marked with **“Supports $filter”** are supported in Microsoft Graph API.

***Values* (Escaping single quotes)**

For requests that use single quotes, if any parameter values also contain single quotes, *they must be double escaped*; otherwise, the request will fail due to invalid syntax.

These limitations are all Microsoft Graph API limitations. Please refer to the Microsoft documentation for details:

https://docs.microsoft.com/en-us/graph/api/overview?view=graph-rest-1.0

https://docs.microsoft.com/en-us/graph/query-parameters

**Paged Read**

**Is Active**

whether (checked) or not (unchecked) paged read mode is used.

**Time Limit**

the time-out value in paged read mode.

**Page Size**

the page size for paged read mode.

**Sorting**

**Sort Attribute**

the sort attribute to be used for paged read mode.

**Sort Order**

the sort order to be used for paged read mode. Valid values are:

**ASCENDING**

sort in ascending order (low values first)

**DESCENDING**

sort in descending order (high values first)



In validation workflows, you should configure a **Sort Attribute** if you configure **Paged Read**. This configuration permits the validation controller to process the entries page by page instead of having to read all pages first into memory before it can start the comparison algorithm.

#### A.10.6.4. Delta

This section describes the parameters needed for delta handling. They either extend the export search filter or set additional operational attributes that are used in the export search request. Both variations are used to retrieve the relevant objects since the last synchronization cycle and produce a delta search result.

**Sort Type**

the method for comparing the relevant attributes. This field is only relevant when **Delta**

**Type** is **SearchAttributes** or **ExpertFilter**. For delta handling, the highest value needs to be calculated by comparing every search result entry with the current highest value, and thus a comparison mode needs to be specified here.

**String**

the relevant attributes (for example, createTimeStamp) are compared as strings.

**Numeric**

the relevant attributes (for example, uSNchanged) are compared as numeric strings.

**Delta Type**

the method for extending the export search.

**SearchAttributes**

the list of attributes to be used; for example, for LDAP:
createTimeStamp
modifyTimeStamp



The export search filter is extended internally with the same definition as listed for **ExpertFilter**.

**ExpertFilter**

an additional SPML filter extension that is combined with export search filter. Example for LDAP:

```
<FilterExtension>
    <dsml:or>
         <dsml:greaterOrEqual name="createTimestamp">
              <dsml:value>${LastDeltaValue}</dsml:value>
         </dsml:greaterOrEqual>
         <dsml:greaterOrEqual name="modifyTimestamp">
              <dsml:value>${LastDeltaValue}</dsml:value>
         </dsml:greaterOrEqual>
    </dsml:or>
</FilterExtension>
```



The notation **${LastDeltaValue}** is mandatory and will be replaced with the highest value calculated by last synchronization cycle.

The filter listed above will also return all the objects starting with the given last delta date.Because "greaterOrEqual" is used, it will return all the objects again that are created or modified within that very last second.Therefore a few updates will be repeated in the next synchronization cycle.If you want to avoid the duplicate entries, you need to use the following filter:

```
<filterExtension>
   <dsml:or>
         <dsml:not>
             <dsml:lessOrEqual name="createTimestamp">
                   <dsml:value>${LastDeltaValue}</dsml:value>
             </dsml:lessOrEqual>
         </dsml:not>
         <dsml:not>
             <dsml:lessOrEqual name="modifyTimestamp">
                   <dsml:value>${LastDeltaValue}</dsml:value>
             </dsml:lessOrEqual>
         </dsml:not>
   </dsml:or>
</filterExtension>
```

**ExpertOpAttributes**

the additional export operational attributes that the connector can use directly for delta handling.Example for ActiveDirectory:

```
<operationalAttributes
         xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core">
     <dsml:attr name="dxm.delta">
              <dsml:valuee type="xsd:base64Binary">
                 ${LastDeltaValue}
              </dsml:value>
     </dsml:attr>
</operationalAttributes>
```



The notation **${LastDeltaValue}** is mandatory and will be replaced with the highest value calculated by last synchronization cycle.Note the notation **dxm.delta** is also mandatory.

#### A.10.6.5. Mapping

This section defines the mapping for this channel.

**Package Name**

the name of the package, if Java mapping is used. It contains part or all of the Java mapping methods.

**Mapping Table**

the mapping definition. See the "Java Mapping Editor" help topic for more information.

#### A.10.6.6. Operational Mapping

This section allows you to define SPML operational attributes that are passed to the connector with every request. They must be defined in correct XML format that is similar to the format for normal attribute mapping, except that the XML element is named "*<operationalAttrMapping>*" instead of "*<attributeMapping>*". Please make sure to include your mapping definitions in a dummy XML root node (just to get well-formed XML) as in the following sample snippet:

```
<?xml version="1.0" encoding="UTF-8"?>
<mappingDefinition>
    <operationalAttrMapping mappingType="constant" name="opx">
        <value>valx</value>
    </operationalAttrMapping>
    <operationalAttrMapping mappingType="constant" name="opy">
        <value>valy</value>
    </operationalAttrMapping>
</mappingDefinition>
```

#### A.10.6.7. Join

This section defines the join definition in XML format. It allows finding the correct entry on the target side. You can define several join expressions that are evaluated in sequence. The join works in these steps:

Evaluate the first join expression.

If the result is exactly one entry, this is the join result.

If the result is zero or more than one entry, use the next join expression (go to 2.).

If there are no more join expressions and the result is zero entries, this is the result.

If more than one entry is found in one of the join expressions and the switch 'Create Despite Multiple Joined Entries' is set, this is the result. Otherwise an error is the result (the join failed).



The join condition for password channels is automatically inherited from the corresponding account channel (specifically, from the channel the Password Primary Channel points to.

For more information about join expressions, see the section about joining.

#### A.10.6.8. Primary Channel

The fields in this tab need to be entered only if secondary channels are to be used. For details on primary and secondary channels, see the chapter "Channels and Mapping" in "Understanding Java-based Workflows" of the *DirX Identity Application Development*

*Guide*. In this case, these fields have to be entered in the secondary channel, which is the one that reflects the database table holding multi-value attributes.

**Primary Channel**

the name of the primary channel representing the entries holding the multi-value attributes.

**Reference Type**

the type of relationship between the primary and the secondary channel. Available options are:
OneToOne - a 1:1 relationship.
ManyToOne - a n:1 relationship.

**Reference from Secondary**

whether (checked) or not (unchecked) the reference in the database is from the table holding the multi-value attributes.

**Join Attributes**

the column name(s) in the database table used for joining the values from the secondary to the primary table.

**Related Topics**

Real-time Activity
Content Tabs
Real-time Filter
Real-time Java Mapping
Java Mapping Editor
Real-time Port
Java-based Workflow

See also "Managing the Java-based Server".

## A.10.7. Content Tabs

These tabs displays the content of this object in XML format, which is the representation used by the Java-based Server as input format.The content definition may contain references that are resolved before the information is loaded into the Java-based Server.Content definitions can also contain other content definitions, which means that they can be structured in a hierarchy.The top-level Content tab contains the complete content after reference resolution.

**Content**

the configuration in XML format.It may contain references to properties of other objects and include statements that reference other content definitions.

**Content (resolved)**

the resolved XML configuration.All references are resolved and all referenced content definitions are contained.
Note: this tab is only visible if design mode is enabled.In some cases - if the content has

no references - the Content (resolved) tab is not available.

**Related Topics**

Real-time Activity
Real-time Channel
Real-time Filter
Real-time Java Mapping
Java Mapping Editor
Real-time Port
Java-based Workflow

## A.10.8. Real-time Java Mapping

A real-time Java source mapping configuration object defines the mapping Java source code.It is a sub-object of a channel configuration object.The configuration data associated with a Java source mapping object includes:

- The object's name, description and version number
- The Java source code and the resulting byte code

Use this tab to enter the properties of a Java source mapping object.The items shown in this tab are:

**Name**
the name of the object.

**Description**
a description of the object.

**Version**
the version number of the workflow.

**Java Source**
the complete Java code for this Java mapping definition. Changing this code and clicking **Save** updates the byte code.

**Byte Code**
the byte code that was calculated from the Java code. This tab is only visible if design mode is enabled.



If you use any third-party libraries in your Java source, be aware that they need to be available both at design time and at runtime, and so they must be deployed in the following locations:

- In the Identity Manager classpath, if they are needed for the compilation. You do this by extending the start file **GUI/bin/dxi_run.bat**.
- In the Java-based Server classpath, in the folder *install_path*/ids-j-***domain***-S*n*/confdb/common/lib*.

**Related Topics**

Real-time Activity
Real-time Channel
Real-time Filter
Real-time Port
Java-based Workflow

See also "Managing the Java-based Server".

## A.10.9. Java Mapping Editor

The Java Mapping Editor is a powerful tool for defining the relationship between source and target attributes.It consists of the mapping table on the left and a tool bar for operating on the table on the right.Use the **Package Name** field above the table to define the package name, if Java mapping is used.It contains all or part of the Java mapping methods.

### A.10.9.1. The Mapping Table

Each row in the mapping table defines the mapping to a target-side attribute.The table provides the following columns:

**Mapping Source**
: the mapping input.The handling of this field depends on the type of mapping (see the next column and the individual mapping types in the tool bar descriptions).

**T(ype)**
: the mapping type.This type is defined during the creation of a new line.Click in the field to change the type from the drop-down list.

**Map to**
: the target system attribute. This field can contain two special values:

**Identifier:*value***
: the first line is reserved for the ID mapping. In this case, a "type" attribute denotes the type of identifier as requested in SPML V1. Open the drop-down list to see the allowed values.

**PostMapping**
: the last line is reserved for the post mapping. This is always a Java mapping (only Java Source or Java Class mapping makes sense). The post mapping is optional and performed after all the attribute mappings. Its Java implementation can work on the entire mapped entry and is especially intended to process dependencies on the target attributes.

**R(ead only)**
: whether (checked) or not (unchecked) the attribute can be read but not written.

**A(dd)**

whether (checked) or not (unchecked) the attribute is only written during add operations. It is not used during modify operations.

**C(heck modification)**

whether (unchecked) or not (checked) the attribute is updated even if it cannot be read at the joined entry because of the amount of data; for example, large groups. Not checking the modification guarantees that the update is always made without comparing existing values. Not checking also means that if an attribute was deleted on the source side and is therefore not contained in the source list of attributes any more, it is not deleted on the target side.

**r(etrievable)**

whether the attribute is readable (checked) or not (unchecked). A good example is a password attribute. You can write it but you can't read it. Uncheck this flag for all attributes that are not retrievable to ensure that they are not passed to the requested attribute list of search and read operations used, for example, during join operations.

**M(odify always)**

whether (checked) or not (unchecked) to add the attribute to the list of modifications if other attributes were changed. If no other attribute was changed, attributes with this flag set are ignored. You can use this mechanism to set, for example, a change status flag if any other attribute was changed. Of course the change status flag should not be set if nothing changes.

**S (notInSchema)**

whether (unchecked) or not (checked) the attribute exists in the schema of the target system in the Identity Store. When this field is checked, the attribute is mapped to the **dxrOptions** attribute when synchronizing data from the target system to the Identity domain.

**e (xact)**

whether (checked) or not (unchecked) the attribute is handled with case-exact matching.

Places where an attribute must be specified:

If you specify a source attribute on the left side of the mapping to map it to a target attribute on the right side, be aware that you must also specify this source attribute in the other direction on the right side to make it readable for the join engine and by that, to be available as a source attribute. For example, if you need an attribute only in one direction - for example, dxmPassword as the source attribute in the target system mapping direction - you must also specify it in the DirX Identity mapping direction. If you only want to read it and not write it to DirX Identity - as in the dxmPassword case - you could insert a direct mapping specifying null on the left side to make it noticeable that it is not filled there and dxmPassword on the right side with the readOnly flag set.

In addition, be aware that if you need an attribute only for specifying it in the export filter, you must also insert a mapping line with this attribute on the right side checked as readOnly.



*Notes on binary attributes:*

To map binary attributes correctly between DirX Identity and a connected system (including a file system), specify the ;binary suffix only for attributes that contain an ASN.1 prefix in their binary data. These attributes have the schema syntax certificate, like the attribute userCertificate, or the schema syntax CrossCertPair or CRL or similar. For attributes that contain only raw binary data without an ASN.1 prefix, which are attributes of schema syntax Octet String, specify the attribute name with the suffix ;raw in the mapping if the attribute does not belong to the standard LDAP attribute schema. If the attribute does belong to the standard LDAP schema - for example, jpegPhoto – a suffix is not required but does not do any harm if specified. If you are unsure whether or not the attribute belongs to the standard LDAP schema, you should specify the ;raw suffix. The suffix - if required - must always be specified in both realtime mapping directions.

#### A.10.9.2. The Tool Bar

The tool bar on the right defines the actions you can perform on the table. The actions you can take depend on the current cursor position in the table. Move the cursor over one of the icons to see tool-tip information.

The actions for insertion of new lines above the cursor line are:

**Constant**

inserts a constant mapping definition. Click in the **Mapping Source** field to open a small window. For single-value attributes, enter the value. Enter each of the required constants for a multi-value attribute into a separate line.

**Direct**

inserts a direct mapping definition. Applies both to single- and multi-value attributes. Double-click in the **Mapping Source** field to enter a source attribute value either directly or via the attribute browser icon at the end of the field. If you use the attribute browser, either use the scroll bar to view the complete list of values or use the text field to enter character by character. Select one of the values in the list and then click **Save** to use this value or click **Cancel** to discard the attribute browser.

**Expression**

inserts a simple expression mapping only for single-value attributes. This field lets you compose strings using simple bean (placeholder) notations. The attribute mapper evaluates the expression at runtime.
You can define several simple expression lines. They are evaluated in sequence during runtime. If a mapping succeeds (that is, it produces a non-null value), it is taken. If not, the join engine tries the next one.
For a more detailed description, see the help topic "Simple Expression Mapping".

**Java Source**

inserts a mapping that is defined via Java source code. Click the New Window button to open a larger window. Enter or edit the source code there. If you click **Apply**, the source code is updated in the small window. If you click Save, the window is closed and the

source code is updated in the small window. If you click Cancel, your edit session is cancelled and all your changes are discarded.
If you save the channel object, all currently uncompiled source code definitions are compiled. Errors are displayed in a separate window. Evaluate the error messages and correct the errors. Click **Save** again to re-check the changed source(s).
The Java class name is built from the attribute name. The first character is converted to uppercase. Illegal characters for a Java class name are replaced with a dollar ($) sign. So the classname to map an attribute **snc.pname** must be **Snc$pname**.
For more details on Java source code mappings see the chapter "Realtime Workflows" in the *DirX Identity Customization Guide*. For sample sources, see also the delivered default workflows.

**Java Class**

inserts a Java class mapping. Double-click in the **Mapping Source** field to enter or edit the class name.
The class name contains the full name of a Java class that is to perform the mapping for the attribute. It must implement the mapping interface applicable for identifier, attribute or post mapping. See the chapter "Realtime Workflows" in the *DirX Identity Customization Guide* for more details.

**Comment**

inserts a comment line. Comment lines are shown as gray lines. Double-click in the **Mapping Source** field to enter or edit the comment. You can enter several lines of comment. Only the first line is visible when the **Mapping Source** field is not selected. If you move the cursor onto a comment line, the complete comment text is shown as a tool tip.

You can select several lines in a mapping table. Use SHIFT to select a range of lines and CTRL to add or remove individual lines to the selection. Note that it is best to select lines in the **Map to** field. The actions for edit operations on the selected lines are:

**Delete**

deletes the selected line(s).

**Duplicate**

duplicates the first selected line.

**Move up**

moves the selected line(s) up one line.

**Move Down**

moves the selected line(s) down one line.

**Copy**

copies the selected line(s) into the clipboard.

**Paste**

pastes the selected line(s) from the clipboard. This works either in the same Manager instance between different mapping tables or between different Manager instances.

### A.10.10. Simple Expression Mapping

This section explains the syntax of simple mapping expressions. You can define one or more expressions, where each is a concatenation of strings:

*expr* [**+** *expr*]

*expr* stands either for a variable or a string constant.

Variable can be:

**${source.***name***}** - source attributes, where *name* defines the attribute name.

**${target.***name***}** - an attribute from the target side (after the attribute is mapped), where *name* defines the attribute name.

**${joinedEntry.***name***}** - attributes from the joined entry in the target system, where *name* defines the attribute name.

**${env.***name***}** - variables taken from the environment, where *name* defines the attribute name.

Note that the following global context environment variables can also be used in simple expression mappings and in user hooks by specifying **${env.***global_var***}**, where *global_var* is evaluated at workflow runtime and can be one of the following:

**dxm.uh.wfName** - the name of the workflow.

**dxm.uh.wfInstID** - the instance ID of the workflow.

**dxm.uh.workDir** - the working directory of the workflow.

**scripts** - the scripts home directory of the Java-based Server.

A string constant is enclosed by double hyphens (for example "constant").

The following limitations apply:

- At least two elements should be defined otherwise the mapping types **Direct** or **Constant** could be used.
- Only single valued attributes are handled.
- If several expressions are concatenated to one string and one of the expressions is empty, the complete expression delivers an empty value.

**Several expression lines:**

You can define several simple expression lines. They are evaluated in sequence during runtime. If a mapping succeeds (that is, it produces a non-empty value), it is taken. If not, the join engine tries the next one.

*Example 11. Single expression line:*

```
${source.dxrName} + ${env.ads_upn_extension}
```
Concatenates the source attribute dxrName and the variable ads_upn_extension from

the environment section.

`${source.sn} + “,” + ${source.initials} + “-” + ${source.givenName}`
Concatenates the source attribute sn, a comma, the source attribute initials, a dash and the source attribute givenName.

*Example 12. Multiple expression lines:*

```
${source.validity}
${env.validity}
```
Uses the source attribute validity if not empty. Otherwise it uses the variable validity from the environment section.

```
${joinedEntry.id}
"cn=" + ${source.ListName} + "," + ${env.role_ts_denygroup_base}
```
If the first line does not retrieve a value (the id of the joined entry is empty, that means there is no joined entry), the second line is evaluated and a new DN is calculated.

```
${source.givenName} + "." + ${source.initials} + "." + ${source.sn}
${source.givenName} + "." + ${source.sn}
${source.sn}
```
If givenName, initials and sn exist, the first line delivers a non-empty result. If one of the values is empty, the second line is evaluated. If the givenName is empty, this line evaluates to an empty result and the last line is evaluated from the sn field. This example assumes that the sn is a mandatory field, thus this multiple definition expression always delivers a value.

### A.10.11. Joining

This section explains how to build join expressions. You can define several join expressions that are evaluated in sequence (from top to bottom). The first expression that delivers a valid result is taken.

You can define lookups (read) or searches.

- Specify lookups via the <searchBase> tag.
- Define searches via the <filterExtension> tag. The search base is taken from the Export tab. The <filterExtension> is combined (by logical AND) with the filter from the Export tab.

Use these definitions to define parameters:

**${source.*attribute*}**
: an attribute from the source side.If you transfer data from the Identity Store to a connected system, then the read entries from the Identity Store are the source entries.If you want to use the identifier of the source entry, then use ${source.id}.

**${target.*attribute*} or ${*attribute*}**

an attribute from the target side (after mapping of this attribute).If you transfer data from the Identity Store to a connected system, then the mapped entries composed from the source information (read from the Identity Store) and the joined entries from the target along with any mapping conversion build the target entries.If you want to use the identifier of the target entry, then use ${target.id}.If you define an attribute without prefix as *${attribute}*, then the join expression is interpreted as ${target.*attribute*}.

***simpleExpression***

an expression that is composed of constants and variables.

**<![CDATA[*simpleExpression*]]>**

an expression that is defined by a simple expression, for example "<GUID=" + ${source.dxmADsGuid} + ">". Use the CDATA construct when you need to escape XML characters like '<' or '>'.

**Joining with IDs (lookup)**

You can use the defined SPML ID for a lookup of exactly this entry:

```
<searchBase type="urn:oasis:names:tc:SPML:1:0#DN">
<spml:id>${source.id}</spml:id>
</searchBase>
```

Alternatively, you can use any source attribute. For a search into an LDAP-based system (such as DirX Identity itself) the value must be of type DN. For a search into another type of connected system, the type depends on the system and the connector:

```
<searchBase type="urn:oasis:names:tc:SPML:1:0#DN">
<spml:id>${source.dxrPrimaryKey}</spml:id>
</searchBase>
```

**Joining with Filter Expressions (Searches)**

Use any attribute (here the sAMAccountName) to search for entries:

```
<filterExtension>
<dsml:equalityMatch name="sAMAccountName">
<dsml:value>${target.sAMAccountName}</dsml:value>
</dsml:equalityMatch>
</filterExtension>
```

The corresponding search base might be defined in the Export tab as:

${env.user_base_rel} + "," + ${env.domain}

You can define AND or OR conditions as defined by the DSML standard:

```
<filterExtension>
<dsml:and>
<dsml:equalityMatch name='externalSystemName'>
   <dsml:value>${target.externalSystemName}</dsml:value>
</dsml:equalityMatch>
<dsml:equalityMatch name='externalDomainName'>
   <dsml:value>${target.externalDomainName}</dsml:value>
</dsml:equalityMatch>
<dsml:equalityMatch name='externalApplicationName'>
   <dsml:value>${target.externalApplicationName}</dsml:value>
</dsml:equalityMatch>
<dsml:equalityMatch name='username'>
   <dsml:value>${target.username}</dsml:value>
</dsml:equalityMatch>
</dsml:and>
</filterExtension>
```

You can use all expressions that are defined in the FilterSet of the DSMLv2 specification as filter conditions, especially **equalityMatch**, **substrings**, **greaterOrEqual**, **lessOrEqual** and **present**. The only pre-requisite is that the associated connector must be able to evaluate them.



Every attribute you use in the filter must have been read with the entry, which means that you need to list it in the mappings even if it will not be updated. In this case, define it as read-only and retrievable and provide an empty mapping expression.

The following examples show more complex join definitions that are used in the delivered default workflows.

*Example 13. Account Joining for ADS Workflows*

```
<joins xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
xmlns:spml="urn:oasis:names:tc:SPML:1:0" >
<join>
<searchBase type="urn:oasis:names:tc:SPML:1:0#GUID">
  <spml:id><![CDATA["<GUID=" + ${source.dxmADsGuid} +
">"]]></spml:id>
</searchBase>
```

```
</join>
<join>
<filterExtension>
  <dsml:equalityMatch name="sAMAccountName">
      <dsml:value>${target.sAMAccountName}</dsml:value>
  </dsml:equalityMatch>
</filterExtension>
</join>
<join>
<searchBase type="urn:oasis:names:tc:SPML:1:0#DN">
   <spml:id>${source.dxrPrimaryKey}</spml:id>
</searchBase>
</join>
</joins>
```

The join process consists of three conditions:

- The first join tries to find the entry via the ADsGuid. This is a special feature of Active Directory.
- Alternatively we use the sAMAccountName of the mapped entry to find the entry.
- If that does not work, we use the primary key (a DN). Note that this definition finds the entry only if it did not move. The first two join definitions find the entry within the whole specified scope.

*Example 14. Account Joining for Imprivata Subscribers*

```
<joins xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
xmlns:spml="urn:oasis:names:tc:SPML:1:0" >
<join>
<filterExtension>
    <dsml:and>
        <dsml:equalityMatch name='externalSystemName'>
            <dsml:value>${target.externalSystemName}</dsml:value>
        </dsml:equalityMatch>
        <dsml:equalityMatch name='externalDomainName'>
            <dsml:value>${target.externalDomainName}</dsml:value>
        </dsml:equalityMatch>
        <dsml:equalityMatch name='username'>
            <dsml:value>${target.username}</dsml:value>
        </dsml:equalityMatch>
    </dsml:and>
```

```
</filterExtension>
</join>
</joins>
```

The join process consists of one condition:

- It is a combined search of three attributes on the target side: externalSystemName, externalDomainName, username.

## A.10.12. Service-Specific Pages

### A.10.12.1. Policy - Job Parameters

Use this tab allows to specify the job parameters when working with provisioning rules.It is not relevant for consistency or validation rules.

**Request Type**

how to process the provisioning rules.Valid options are:

**Process Rules**

evaluate and process the rules.

**Simulate Rules**

evaluate and simulate the rules.

**Provisioning Mode**

the mode in which the rule engine works when resolving privileges.Valid options are:

**Assign Privilege and Resolve**

assigns all privileges of the current rule to each user and resolves the user afterwards immediately.

**Assign Privilege Only**

assigns all privileges of the current rule to each user but does not resolve it; instead, the TBA flag is set and you must run a privilege resolution after this run of the policy execution.Note this option does not work if separation of duties (SoD) is enabled because SoD evaluation requires an immediate privilege resolution for each user.

**Suppress Change Events**

whether (unchecked) or not (checked) to initiate user change events.

**Synchronous Resolution**

if checked, the controller will resolve a user immediately (synchronous). Otherwise (unchecked), it will send a resolve message so that a resolution adapter in a Java server will resolve the user later (asynchronous)

**Enabling prefetching privileges into cache**

whether (checked) or not (unchecked) privileges should be loaded into the cache before finding the matching groups during resolution. It can only be checked if *Synchronous Resolution* is checked. This is useful for performance reasons, but be aware that it can cause issues if the system does not have enough configured memory to store the prefetched objects.

**Related Topics**

Policy - Rule Search Parameters
Policy - Object Search Parameters

### A.10.12.2. Policy - Rule Search Parameters

Use this tab to define which rules to process and set some optimization parameters.

**Base Object**

the base node at which to start the search for rules.

**Subset**

the method used to perform the search.Allowed values are:

**BASEOBJECT**

retrieve only the base object where the search started.

**ONELEVEL**

retrieve all objects from the next level below the base object.

**SUBTREE**

retrieve the complete set of objects from the subtree.

**Filter**

an optional filter that retrieves specific object types. Typical filters are:

**(&(objectClass=dxrPolicy)(dxrType=ProvisioningRule))**

retrieves provisioning and validation rules.

**(&(objectClass=dxrPolicy)(dxrType=ConsistencyRule))**

retrieves only consistency rules.

**(&(objectClass=dxrPolicy)(dxrType=ValidationRule))**

retrieves only validation rules.

**Sort Key**

the sort criteria (default is cn). This parameter allows you to define a sequence of how the consistency and validation rules are processed. It is not used for provisioning rules.

The next two parameters allow for rule processing optimization. All rules and operations are loaded into a storage layer cache.

**Rule LDAP Page Size**

size of a result page for paged searches (number of rules; default: 300).

**Rule Cache MRU Size**

the size of the most recently used cache for objects being read via LDAP (default: 500).The Rule Cache MRU Size defines the size of the most recently used cache in the storage layer that holds the objects that are not affected by the garbage collector.If the Rule Cache MRU Size is too low, the objects are removed by the garbage collector and thus must be restored from LDAP when they are accessed the next time, which slows down the agent significantly.We recommend that you set this value to twice the number of rules that are read (the corresponding operations are also cached).

For more information on how to use these optimization parameters, see the section "Using the Maintenance Workflows → Understanding the Tcl-based Maintenance Workflows → Privilege Resolution Workflow → Privilege Resolution Workflow Optimization" in the DirX Identity Application Development Guide.

**Related Topics**

Policy - Job Parameters
Policy - Object Search Parameters

### A.10.12.3. Policy - Object Search Parameters

Use this tab to define all parameters to search the set of users and the related objects (accounts, groups, roles, permissions, assignments, target systems).These parameters include:

**Object LDAP Page Size**

the size of a result page for paged searches (default: 50).

**Object Cache**

whether (checked) or not (unchecked) the object cache is enabled (default: TRUE).

**Object Cache MRU Size**

the objects being read via LDAP are stored in a cache in the storage layer.The Object Cache MRU Size defines the size of the most recently used cache, holding the objects that are not affected by the garbage collector.So if we have 300 Users in the prefetch cache where each of them has 10 accounts and 10 groups assigned, this results in 6300 objects being stored in the cache (roles, permissions, ...).If the Object Cache MRU Size is too low, the the garbage collector removes the objects and they must subsequently be restored from LDAP when they are accessed the next time.This action slows down the agent considerably.

**Object Accumulator Size**

controls the algorithm that calculates the affected users from the privileges to be analyzed.Increasing this value reduces the number of LDAP searches but increases the length of the search filters.

For more information on how to use these parameters, see the section "Using the

Maintenance Workflows → Understanding the Tcl-based Maintenance Workflows → Privilege Resolution Workflow → Privilege Resolution Workflow Optimization" in the *DirX Identity Application Development Guide*.

**Related Topics**

Policy - Job Parameters
Policy - Rule Search Parameters

#### A.10.12.4. Service - Job Parameters

Use this tab to define the mode in which the service agent runs.

**Request Type**

the mode in which the service agent runs.Possible values are:

**Check Consistency**

run in consistency mode.This comprises a set of hard programmed consistency rules.For details, see the section "Using the Maintenance Workflows → Understanding the Tcl-based Maintenance Workflows → Consistency Check Workflow → Consistency Check Workflow Operation" in the *DirX Identity Application Development Guide*.You can also set up consistency rules that run in the policy execution workflow.

**Resolution**

run a set of consistency checks and then resolves the defined user set.For details, see the section "Using the Maintenance Workflows → Understanding the Tcl-based Maintenance Workflows → Privilege Resolution Workflow → Privilege Resolution Workflow Operation" in the *DirX Identity Application Development Guide*.

**User Resolution**

resolve the defined user set only.For details, see the section "Using the Maintenance Workflows → Understanding the Tcl-based Maintenance Workflows → Privilege Resolution Workflow → Privilege Resolution Workflow Operation" in the *DirX Identity Application Development Guide*.

**Generate Report**

run in report generation mode. Set the report size limit field accordingly.

**Suppress Change Events**

whether (unchecked) or not (checked) to initiate user change events.

**Synchronous Resolution**

if checked, the controller will resolve a user immediately (synchronous). Otherwise (unchecked), it will send a resolve message so that a resolution adapter in a Java server will resolve the user later (asynchronous)

**Enabling prefetching privileges into cache**

whether (checked) or not (unchecked) privileges should be loaded into the cache before finding the matching groups during resolution. It can only be checked if *Synchronous Resolution* is checked. This is useful for performance reasons, but be aware that it can

cause issues if the system does not have enough configured memory to store the prefetched objects.

**Report Size Limit**

the maximum number of records to process in a report.

**Related Topics**

Service - Import Properties
Service - Limits
Service - Requested Attributes

Service - Import Properties
Service - Limits
Service - Requested Attributes

### A.10.12.5. Service - Import Properties

Use this tab to define the import properties for the service agent (used for consistency checking, privilege resolution and report generation):

**Subject Filter**

the filter for user entries to be processed.The default filter is:
(objectClass="dxrUser" and dxrTBA="True")
You can define with this filter whether the agent shall process all users, only the flagged ones (dxrTBA=TRUE) or a subset that is specified with other attributes (for example, (ou="Finance")).

**Related Topics**

Service - Limits
Service - Job Parameters
Service - Requested Attributes

### A.10.12.6. Service - Requested Attributes

Use this tab to define the attributes to read for rules.

**Attribute Configuration**

the attributes that are available.Select an attribute and then use the buttons in the middle of the display to add it to the selected attributes.

**Selected Attributes**

the attributes that are currently selected.Select an attribute and then use the buttons in the middle of the display to remove it from the selected attributes.

**Related Topics**

Service - Import Properties
Service - Limits
Service - Job Parameters

#### A.10.12.7. Service - Limits

Use this tab to define the optimization parameters for the service agent (which is used for consistency checking, privilege resolution and report generation).

The service agent uses an object cache that handles users, accounts, groups, roles, permissions, assignments and target systems.

**Object LDAP Page Size**

the size of a result page for paged searches.

**Object Cache MRU Size**

the size of the most recently used object cache.Objects read via LDAP are stored in a cache in the storage layer.The Object Cache MRU Size defines the size of the most recently used cache that holds the objects that are not affected by the garbage collector.If the Object Cache MRU Size is too low, the garbage collector removes the objects and they subsequently must be restored from LDAP when they are accessed the next time, which slows down the agent considerably.

**Object Accumulator Size**

controls the algorithm that calculates the affected users from the privileges to be analyzed.Increasing this value reduces the number of LDAP searches but increases the length of the search filters.

**Related Topics**

Service - Import Properties
Service - Job Parameters
Service - Requested Attributes

### A.10.13. Transport Workflows

#### A.10.13.1. Transport - Connection

Use this tab to define the connection parameters for the directory server.

**Connected Directory**

the directory where the information is to be retrieved or stored.

**Bind Profile**

the bind information, typically user and password.

For more information, see the section "Using Utilities → Transporting Data" in the *DirX Identity User Interfaces Guide*.

**Related Topics**

Content Tabs
Transport - Export
Transport - Import

#### A.10.13.2. Transport - Export

Use this tab to define the parameters for the object collection export.It allows you to export a collection list, a filter with a search base or a combination of both.

**Filter**

**Search Base**

the node at which to start the search for collections.

**Filter**

the filter to use to restrict the search.This filter must define objects of type dxmCollection.



you can use the **Filter** condition or the **Collections** definition, but not both!

**Collections**

**List of Collections**

the list of object collections.



you can use the **Filter** condition or the **Collections** definition, but not both!

**Format**

**Base64 Enabled**

whether (checked) or not (unchecked) the output of special characters is performed in Base64 format (standard LDIF format). If the flag is not set, a readable format is generated, which is especially useful if you want to store the files in a configuration management system that calculates differences.

**Max. Line Length**

the maximum line length after which the lines are wrapped around. If set to 0 (the default), the lines are not wrapped.

**Page Size**

the page size for internal LDAP searches.If set to 0 (the default), paging is not performed.

For more information, see the section "Using Utilities → Transporting Data" in the *DirX Identity User Interfaces Guide*.

**Related Topics**

Content Tabs
Transport - Connection
Transport - Import

#### A.10.13.3. Transport - Delete

Use this tab to define the parameters for an optional object collection deletion that is

performed before the import operation.Note that the collection definitions are taken from the target, which means that they reflect the set of previously imported entries.

**Delete entries before importing new ones**

**Is active**

whether (checked) or not (unchecked) to delete all entries defined by the list of collections specified in **Collections**.

**Collections**

a list of collection definitions in the target directory.These collection definitions typically reflect the set of entries that were imported during the last run of this workflow.

**Related Topics**

Content Tabs
Transport - Connection
Transport - Import

#### A.10.13.4. Transport - Import

Use this tab to define the parameters for the object collection import.Import handles previously exported collection files defined as a file list and provides custom mapping of attributes.

**Import**

**Files**

the list of files to import.

**SPML**

**Enabled**

whether (checked) or not (unchecked) to define SPML format for the input files.

**Validate**

whether (checked) or not (unchecked) to check for correct SPML format. Using this option requires some extra time.

**LDIF**

**Binary Attributes (comma separated)**

the list of attributes with binary format.

**Input Filter**

the filter condition for the import operation. For example, you can import only objects of a specific object class.

**Simulation Mode**

**none (default)**
: modifies the target.

**loggerLDIF**
: records LDIF requests and responses and modifies the target.

**loggerSPML**
: records SPML request and responses and modifies the target.

**simulateLDIF**
: records LDIF requests and responses but does not modify the target.

**simulateSPML**
: records SPML request and responses but does not modify the target.

**Log Filename**
: the location (path and filename) to which the log file is written.

You can also specify one or more domain mappings and a set of transport attribute mappings that influence the import operation.

For more information, see the chapter/section "Using Utilities → Transporting Data" in the *DirX Identity User Interfaces Guide*.

**Related Topics**

Transport - Attribute Configuration Old
Transport - Attribute Configuration New
Content Tabs
Transport - Connection
Transport - Export
Transport - Delete
Transport - Domain Mapping

#### A.10.13.5. Transport - Domain Mapping

Use this tab to define domain mappings for easy conversion of domain specifications.

**Domain Mapping**

**Domain Mappings**
: a list of domain mapping specifications.Define the source and target domain for each mapping.

*Example 15. Domain Mapping*

```
Source: cn=My-Company Target: cn=AnyDomain
Source: cn=* Target: cn=DefaultDomain

This definition converts data exported from the My-Company domain to a
```

Provisioning domain with the name "AnyDomain".Data exported from all other domains are imported to the "DefaultDomain".Without the cn=* line for data from other domains no domain mapping takes place.

For more information, see the chapter/section "Using Utilities → Transporting Data" in the *DirX Identity User Interfaces Guide*.

**Related Topics**

Transport - Attribute Configuration Old
Transport - Attribute Configuration New
Content Tabs
Transport - Connection
Transport - Delete

### A.10.13.6. Transport - Attribute Configuration New

Use this tab to enter the properties of a transport attribute configuration.The items shown in this tab are:

**Name**
the name of the object.

**Description**
a description of the object.

**Type of Definition**
the mapping definition type.

**Attribute Match**
the pattern to use to match the mapped attribute name.



You need to create some mapping choices by selecting **New Mapping Choice** from the context menu of the attribute configuration node.

**Related Topics**

Transport - Attribute Mapping Choice
Content Tabs
Transport - Connection
Transport - Delete
Transport - Domain Mapping

### A.10.13.7. Transport - Attribute Configuration Old

Use this tab to modify the properties of an old-style transport attribute configuration and mapping.The items shown in this tab are:

**Name**
the name of the object.

**Description**
a description of the object.

**Type of Definition**
the mapping definition type.

**Attribute Match**
the pattern to use to match the mapped attribute name.

**Operation**
the mapping type to be used

**Has one of these**
objects that have one of the listed object classes match. Define the classes as a blank-separated list.

**Has all of these**
only objects that have all of the listed object classes match. Define the class list as a blank-separated list.

You can modify existing old-style attribute configurations that influence the import data stream. Note that it is no longer possible to create old-style Transport Attribute Configuration elements below the import workflow's "perform" activity. The first field defines the type of definition. Select from the list. The following definition types exist:

**ConstantAttributeDefinition**

Use this definition type to set attributes to a constant value.

**Constant Values** - a single value for single value attributes, multiple values for multi-value attributes.

**ExpressionAttributeDefinition**

Use this definition type to set attributes to a value that is calculated via a simple expression.

**Expression Value**
the expression value as a simple expression. See the topic "Simple Expression Mapping" for more information.

**JavaClassAttributeDefinition**

Use this definition type to set attributes to a value that is calculated via a Java class.

**Java Class**
the Java class that defines the transformation. See the topic "Java Mapping" for more information.

#### ReplacementAttributeDefinition

Use this definition type to replace parts of an attribute value.

**Replace Pattern**

a replacement pattern in the format:

*flags/pattern/replacement*

where:

*flags* - flags in the form [afmi] {0,2}:

**a** - replace all (default)

**f** - replace first

**m** - match case (default)

**i** - ignore case

*pattern* - Java regular expression pattern syntax.

*replacement* - the replacement string.



Replacements do not work on the distinguished name attribute (DN). Use the domain mapping to replace the domain part of the DN.

Examples:

This definition replaces the first occurrence of the exact string "here" to "in this field".

```
f/here/in this field/
```

The following specification replaces all occurrences of "Add" or "add" or "ADD" etc. to "Put".

```
i/Add/Put/
```

#### StandardAttributeDefinition

Use this definition type to define an operation to be applied to this attribute type.

**Operation**

the operation to perform. The following operations are available:

**delete**

delete this attribute.

**direct**
: set the attribute value to the value of the attribute in the input file. This is the default operation if no mapping is defined for an attribute.

**exclude**
: do not modify this attribute.

**onaddonly**
: set this attribute only during add operations (the first time the object is transferred).

**onemptyonly**
: set this attribute only if its value is empty and if the entry already exists.

For more information, see the chapter/section "Using Utilities → Transporting Data" in the *DirX Identity User Interfaces Guide*.

**Related Topics**

Transport - Attribute Configuration New
Content Tabs
Transport - Import
Transport - Domain Mapping

#### A.10.13.8. Transport - Attribute Mapping Choice

Use this tab to specify attribute mappings that influence the import data stream.Create Transport Attribute Mapping Choice elements below the Transport Attribute Config element of the import workflow's "perform" activity.After creation, you can specify the mapping in detail.

**General**

This section provides general information about the mapping choice.Available fields are:

**Name**
: the name of the mapping choice.

**Description**
: a description of the mapping choice.

**Type of Mapping**
: the mapping type to be used. It determines the available configuration fields. The mapping types are described in the **Mapping** section.

**When Applicable**

This section defines the matching conditions that must be met in order to perform a given mapping on the imported entry. Available fields are:

**Mapping flags**

flags that restrict the use of a given mapping.

**onaddonly**

the imported entry is matched only for add operations (the first time the object is transferred).

**onemptyonly**

the imported entry is matched only if its existing value is empty and the target entry already exists.

**DN patterns**

the list of patterns to be matched with the imported entry DN. At least one must match.

**Filter**

the LDAP filter to be matched with imported entry attributes.

**Mapping**

The available mapping types are:

**ConstantAttributeDefinition**

Use this definition type to set attributes to a constant value.

**Constant Values**

a single value for single value attributes, multiple values for multi-value attributes.

**ExpressionAttributeDefinition**

Use this definition type to set attributes to a value that is calculated via a simple expression.

**Expression Value**

the expression value as a simple expression. See the topic "Simple Expression Mapping" for more information.

**JavaClassAttributeDefinition**

Use this definition type to set attributes to a value that is calculated via a Java class.

**Java Class**

the Java class that defines the transformation. See the topic "Java Mapping" for more information.

**ReplacementAttributeDefinition**

Use this definition type to replace parts of an attribute value.

**Replace Pattern**

a replacement pattern in the format:

*flags/pattern/replacement*

where:

*flags* - flags in the form [afmi]\{0,2}:
**a** - replace all (default)

**f** - replace first

**m** - match case (default)

**i** - ignore case

- *pattern* - Java regular expression pattern syntax.
- *replacement* - the replacement string.



Replacements do not work on the distinguished name attribute (DN). Use the domain mapping to replace the domain part of the DN.

Examples:

The following definition replaces the first occurrence of the string "Here" to "in this field":

```
f/Here/in this field/
```

The following specification replaces all occurrences of "Add" or "add" or "ADD" and son on to "Put":

```
ai/Add/Put/
```

**DeleteAttributeDefinition**

Delete the attribute value in target entry.

**ExcludeAttributeDefinition**

Do not modify this attribute value.

**DirectAttributeDefinition**

Set the attribute value to the value of the attribute in the imported entry. This is the default mapping operation if no mapping is defined or matched for an attribute.

For more information, see the chapter/section "Using Utilities → Transporting Data" in the

*DirX Identity User Interfaces Guide*.

**Related Topics**

Transport - Attribute Configuration New
Content Tabs
Transport - Connection
Transport - Import

## A.10.14. Workflow-Specific Pages

### A.10.14.1. Campaign Generator Workflows

#### A.10.14.1.1. Campaign Generator - Initiator

Use this tab to define initiators (responsible persons) for multiple campaign generator jobs.The available parameters are:

**Initiator**

the distinguished name of a user that acts as the initiator of the campaign generator workflow.This value is also used to start the certification workflows and is included in audit messages if audit is enabled.If you do not enter a user here, the technical user (the DomainAdmin) is used as the initiator.

**Password**

the password of the initiator.This field is optional.If you enter a password, the campaign generator authenticates with this user / password combination.If the field is empty, it simply sets the initiator value.

**Group Users by Attribute**

Enter the name of a special "grouping" attribute here if you want to start a separate certification workflow for each of the groups.For example if you enter **manager** here, all users in a certification workflow will have the same manager.

**Initiator from Grouping Attribute**

If checked, each workflow is started with the user referenced by the grouping attribute as initiator.Note that it only makes sense to check this flag if the attribute is a DN reference to a user.By this mechanism, it is easy to define for example the user's manager as the approver of the workflow.

Users with an empty grouping attribute are gathered in a separate group having the person entered as **Initiator** in the **Campaign Initiator** step as the certification workflow's initiator.

**Maximum Number of Users per Workflow**

Entering a number here limits the number of users per certification workflow. This setting must be used to assure certification workflows can be handled by the system. A setting of 100 is suitable, with 1000 users per workflow the IdS-J is likely to run into a timeout when preparing the approval data for Web Center.

The initiator is used for authentication and auditing.

**Related Topics**

Campaign Generator - Groups
Campaign Generator - Permissions
Campaign Generator - Roles

#### A.10.14.1.2. Campaign Generator - Roles

Use this tab to specify the roles for this access certification campaign.For more information about possible scenarios for access certification, see the corresponding use case document.

The available parameters are:

**Process Roles**

whether (checked) or not (unchecked) role processing is enabled.You can set up all of the parameters in this tab and then use this flag to control whether or not they're used.

**Process Indirect Assignments**

whether (checked) or not (unchecked) to process indirect assignments; that is, assignments by rules or business object inheritance.By default, the campaign generator creates certification workflows that contain for each privilege all users that have this privilege manually assigned.If you set this flag, the workflow also contains all users that have this privilege assigned by rules or business object inheritance.Note that only the manually-assigned users can be certified; the ones who are automatically assigned are only shown for information.If a privilege has no manually-assigned roles, no certification workflow is started.

**Defaults**

**Certification Workflow**

the workflow to start for certification. If nothing is specified in the corresponding field at the privilege, this workflow is used by default.

**Certification Period**

the time period between certifications if nothing is specified in the corresponding field at the privilege. For example, suppose a certification takes place on the 1st of June. When the certification period is set to 6 months, the next certification will take place on the 1st of November.

**Role Search**

The roles for which an access certification is to be started. The available fields are:

**Role Search Base**

the search base at which to start the role search. If nothing is specified, the entire role catalog is used.

**Role Filter**

the filter to use to search for roles. If nothing is specified, the following default filter is used:
(objectClass="dxrRole" and dxrNeedsCertification="true" and (dxrCertificationPending="FALSE" or not (dxrCertificationPending=**))**
**and (not (dxrState="DELETED") or not (dxrState=**)) and dxrCertificationDate ⇐"$(gmtime)")
It searches for all roles that have the certification flag set, where the certification date is passed and where no certification workflow is already running. Furthermore deleted roles are ignored.

**Page Size**

the page size for the role search. Because the creation of certification workflows can be time-consuming (especially when there are many users are to certify) you should set low values in this field. If the paging time-out of the LDAP server is reached, use a smaller value.

**User Search**

The users to be certified by this campaign. You can use these parameters to divide the campaign; for example, to run campaigns for each country or to run campaigns for specific organizational parts. The available fields are:

**User Search Base**

the search base at which to start the user search. You can separate the campaign into a set of trees with different persons to approve. If nothing is specified, the entire user tree is searched for affected users.

**Additional User Filter**

an additional filter to define the set of users for this campaign.If empty, all users are considered.

**Page Size**

the page size for the user search.

**Related Topics**

Campaign Generator - Groups
Campaign Generator - Initiator
Campaign Generator - Permissions

#### A.10.14.1.3. Campaign Generator - Permissions

Use this tab to specify the permissions for this access certification campaign.For more information about possible scenarios for access certification, see the corresponding use case document.

The available parameters are:

**Process Permissions**

whether (checked) or not (unchecked) permission processing is enabled.You can set up all of the parameters in this tab and then use this flag to control whether or not they are used.

**Process Indirect Assignments**

whether (checked) or not (unchecked) to process indirect assignments; that is, assignments by rules or business object inheritance.By default, the campaign generator creates certification workflows that contain for each privilege all users that have this privilege manually assigned.If you set this flag, the workflow also contains all users that have this privilege assigned by rules or business object inheritance.Note that only the manually-assigned users can be certified; the ones who are automatically assigned are only shown for information.If a privilege has no manually-assigned permissions, no certification workflow is started.

**Defaults**

**Certification Workflow**

the workflow to start for certification. If nothing is specified in the corresponding field at the privilege, this workflow is used by default.

**Certification Duration**

the default time period for the approver(s) to certify the permissions, if nothing is specified in the corresponding field at the privilege. Typically you should set this period to about 4 weeks to give the approvers enough time to complete the certification.

**Certification Period**

the time period between certifications, if nothing is specified in the corresponding field at the privilege. For example, suppose a certification takes place on the 1st of June. When the certification period is set to 6 months, the next certification will take place on the 1st of November.

**Permission Search**

The permissions for which an access certification is to be started. Available fields are:

**Permission Search Base**

the search base where to start the permission search. If nothing is specified, the whole set of permissions is used.

**Permission Filter**

the filter to search for permissions. If nothing is specified, the default filter is used:
(objectClass="dxrPermission" and dxrNeedsCertification="true" and (dxrCertificationPending="FALSE" or not (dxrCertificationPending=**))**
**and (not (dxrState="DELETED") or not (dxrState=**)) and dxrCertificationDate ⇐"$(gmtime)")
It searches for all permissions that have the certification flag set, where the certification date is passed and where no certification workflow is already running. Furthermore deleted permissions are ignored.

**Page Size**

the page size for the permission search. Because the creation of certification workflows can be time-consuming (especially when there are many users are to certify) you should set low values. If the paging time-out of the LDAP server is reached, use a smaller value.

**User Search**

The users to be certified by this campaign. You can use these parameters to divide the campaign; for example, to run campaigns for each country or to run campaigns for specific organizational parts. Available parameters are:

**User Search Base**

the search base at which to start the user search. You can separate the campaign into a set of trees with different persons to approve. If nothing is specified, the entire user tree is searched for affected users.

**Additional User Filter**

an additional filter to define the set of users for this campaign.If empty, all users are considered.

**Page Size**

the page size for the user search.

**Related Topics**

Campaign Generator - Groups
Campaign Generator - Initiator
Campaign Generator - Roles

#### A.10.14.1.4. Campaign Generator - Groups

Use this tab to specify the groups for this access certification campaign.For more information about possible scenarios for access certification, see the corresponding use case document.

The available parameters are:

**Process groups**

whether (checked) or not (unchecked) group processing is enabled.You can set up all parameters in this tab and then use this flag to control whether or not they’re used.

**Process Indirect Assignments**

whether (checked) or not (unchecked) to process indirect assignments; that is, assignments by rules or business object inheritance.By default, the campaign generator creates certification workflows that contain for each privilege all users that have this privilege manually assigned.If you set this flag, the workflow also contains all users that have this privilege assigned by rules or business object inheritance.Note that only the manually-assigned users can be certified; the ones who are automatically assigned are only shown for information.If a privilege has no manually-assigned groups, no certification workflow is started.

### Defaults

#### Certification Workflow

the workflow to start for certification. If nothing is specified in the corresponding field at the privilege, this workflow is used by default.

#### Certification Duration

the default time period for the approver(s) to certify the groups if nothing is specified in the corresponding field at the privilege. Typically you should set this period to about 4 weeks to give the approvers enough time to complete the certification.

#### Certification Period

the period between certifications if nothing is specified in the corresponding field at the privilege. For example, suppose a certification takes place on the 1st of June. When the certification period is set to 6 months, the next certification will take place on the 1st of November.

### Group Search

The groups for which an access certification is to be started. Available parameters are:

#### Group Search Base

the search base at which to start the group search. If nothing is specified, the entire set of groups is used.

#### Group Filter

the filter to use to search for groups. If nothing is specified, the following default filter is used:
(objectClass="dxrTargetSystemGroup" and dxrNeedsCertification="true" and (dxrCertificationPending="FALSE" or not (dxrCertificationPending=**))**
**and (not (dxrState="DELETED") or not (dxrState=**)) and dxrCertificationDate ⇐"$(gmtime)")
It searches for all groups that have the certification flag set, where the certification date is passed and where no certification workflow is already running. Deleted groups are ignored.

#### Page Size

the page size for the group search. Because the creation of certification workflows can be time-consuming (especially when there are many users are to certify) you should set low values. If the paging time-out of the LDAP server is reached, use a smaller value.

### User Search

The users to be certified by this campaign. You can use these parameters to divide the campaign; for example, to run campaigns for each country or to run campaigns for specific organizational parts. Available parameters are:

#### User Search Base

the search base at which to start the user search. You can separate the campaign into a set of trees with different persons to approve. If nothing is specified, the entire user tree

is searched for affected users

**Additional User Filter**

an additional filter to define the set of users for this campaign. If empty, all users are considered.

**Page Size**

the page size for the user search.

**Related Topics**

Campaign Generator - Initiator
Campaign Generator - Permissions
Campaign Generator - Roles

#### A.10.14.2. Consistency Check Workflows

This tab is displayed with the main activity of a consistency check workflow. The available fields depend on the workflow (UserResolution, MarkAffectedUsers, CheckConsistency).

##### A.10.14.2.1. User Resolution Workflow

Use this tab to set up Consistency Check User Resolution workflow processing.

The following field defines the Provisioning rules to be evaluated.

**Search Base for Provisioning Rules**

evaluate all Provisioning rules in the specified subtree.

The following fields allow you to define the users to be processed:

**Search Base for Users**

process only those users in the specified subtree.

**Filter for Users**

process only those users that match the specified filter and are in the specified sub tree. When this field is empty, the workflow checks all users in the specified subtree.

**Use Filter for Users directly (don't 'AND' with dxrtba=true)**

whether (true) or not (false) the defined user filter is used as it is or modified if **dxrtba** is set. For example, you can define a filter that processes all users regardless of whether or not **dxrtba** is set. If not set or false, the defined user filter is implicitly ***and*ed** with (&(objectClass=dxrUser)(dxrTBA=TRUE)(dxrState=*). This feature was introduced in DirX Identity V8.9. If you have workflows created in older versions, this feature may not work out of the box because the stored XML content does not contain this property. In this case, you can update the content by clicking **Edit**, switching the controller to the **MarkAffectedUserController**, switching back to the **PrivilegeResolutionController** and then clicking **Save**. This action updates the XML content and after a "Load IdS-J Configuration" command, you can use this new field.

**Optimization Parameters**

There are two sets of optimization parameters: the first set controls the application of provisioning rules and the second set controls the user resolution process.

Use the following parameters to optimize the application of Provisioning rules:

**LDAP Page Size for Provisioning Rules**
: the size of a result page for paged searches (number of rules; default: **300**).

**Cache MRU Size for Provisioning Rules**
: the size of the most recently used cache for objects read via LDAP (default: **500**). The Rule Cache MRU Size defines the size of the most recently used cache in the storage layer that holds the objects that are not affected by the garbage collector. If the Rule Cache MRU Size is too low, the garbage collector removes the objects and they must subsequently be restored from LDAP when they are accessed the next time, which slows down the agent significantly. We recommend that you set this value to twice the number of rules that are read (the corresponding operations are also cached).

**Batch Accumulator Size for Provisioning Rules**
: controls the algorithm that calculates the affected users from the privileges to be analyzed. Increasing this value reduces the number of LDAP searches but increases the length of the search filters.

**Use User Cache**
: whether or not to enable the object cache (default: TRUE).

Use the following parameters to optimize the user resolution process:

**LDAP Page Size for User Resolution**
: the size of a result page for paged searches.

**Cache MRU Size for User Resolution**
: the size of the most recently used object cache. Objects that are read via LDAP are stored in a cache in the storage layer. The Object Cache MRU Size defines the size of the most recently used cache that holds the objects that are not affected by the garbage collector. If the Object Cache MRU Size is too low, the garbage collector removes the objects and they must subsequently be restored from LDAP when they are accessed the next time, which slows down the workflow considerably.

**Batch Accumulator Size for User Resolution**
: controls the algorithm that calculates the affected users from the privileges to be analyzed. Increasing this value reduces the number of LDAP searches but increases the length of the search filters.

##### A.10.14.2.2. Mark Affected Users Workflow

Use this tab to optimize the Consistency Check Mark Affected Users workflow. Optimization parameters include:

**LDAP Page Size**

the size of a result page for paged searches (default: **300**).

**Cache MRUS ize**

the size of the most recently used cache for objects read via LDAP (default: **500**). The Rule Cache MRU Size defines the size of the most recently used cache in the storage layer that holds the objects that are not affected by the garbage collector. If the Rule Cache MRU Size is too low, the garbage collector removes the objects and they must subsequently be restored from LDAP when they are accessed the next time, which slows down the agent significantly. We recommend that you set this value to twice the number of rules that are read (the corresponding operations are also cached).

**Batch Accumulator Size**

controls the algorithm that calculates the affected users from the privileges to be analyzed. Increasing this value reduces the number of LDAP searches but increases the length of the search filters.

#### A.10.14.2.3. Consistency Check Workflow

Use this tab to set Consistency Check workflow (CheckConsistency) attributes.

**Optimization Parameters**

**LDAP Page Size**

the size of a result page for paged searches (default: **300**).

**Cache MRU Size**

the size of the most recently used cache for objects read via LDAP (default: **500**). The Rule Cache MRU Size defines the size of the most recently used cache in the storage layer that holds the objects that are not affected by the garbage collector. If the Rule Cache MRU Size is too low, the garbage collector removes the objects and they must subsequently be restored from LDAP when they are accessed the next time, which slows down the agent significantly. We recommend that you set this value to twice the number of rules that are read (the corresponding operations are also cached).

**Batch Accumulator Size**

controls the algorithm that calculates the affected users from the privileges to be analyzed. Increasing this value reduces the number of LDAP searches but increases the length of the search filters.

**Check for Privileges To Be Deleted**

whether (checked) or not (unchecked) privileges to be deleted are checked. When checked, the workflow performs the following actions:

- Searches for roles, permissions, and groups in the state TBDEL.
- For each privilege in the state TBDEL, the workflow:
  - Removes the incoming assignments from users and/or senior privileges and sets the **To Be Analyzed** (TBA) flag for the affected objects.

- Deletes roles/permissions or sets their state to DELETED if history is configured.
- Sets the state of groups to DELETED.

**Check Users**

**Check Users**

whether (checked) or not (unchecked) a set of specified users should be checked. When checked, the following parameters define the users to be checked:

**Search Base**

the search base for users to be checked.

**Filter**

the filter for users to be checked.

**Check Roles/Permissions**

**Check Roles and Permissions**

whether (checked) or not (unchecked) a specified set of roles and permissions should be checked. When checked, the following parameters define the roles and permissions to be checked:

**Search Base for Roles**

the search base for roles to be checked.

**Filter for Roles**

the filter for roles to be checked.

**Search Base for Permissions**

the search base for permissions to be checked.

**Filter for Permissions**

the filter for permissions to be checked.

**Check Accounts/Groups**

**Check Accounts and Groups**

whether (checked) or not (unchecked) the accounts and groups in a set of specified target systems should be checked. When checked, the following parameters define the target systems to be checked:

**Search Base for Target Systems**

the search base for the target systems.

**Filter for Target Systems**

the filter for the target systems.

**Apply Consistency Rules**

**Apply Consistency Rules**

- whether (checked) or not (unchecked) to apply a specified set of consistency rules. When checked, the following parameters specify the consistency rules to be applied and provide optimization options for them:

**Search Base**

the search base for consistency rules.

**Filter**

the filter for consistency rules.



For assignment checks, the workflow skips already active assignments whose start date has already been reached unless they are in state **FUTURE** and their start date has been reached. Only assignments whose end date has been reached or whose state is **FUTURE** and whose start date has been reached are processed. This avoids repeated **dxrTBA** writes for assignments that do not require any change. Make sure that assignments with a future start date are always stored with **dxrState=FUTURE**.

**Sort Key**

Use this optional field to change the search attribute with which the workflow searches rules. The default attribute is **cn**.

**Sort Ascending**

Use this optional field to change the search order in which the workflow searches rules. The default is ascending **TRUE**.

**LDAP Page Size**

the size of a result page for paged searches (applies only to the specified consistency rules) (default: **300**).

**Cache MRU Size**

the size of the most recently used cache for objects read via LDAP (applies only to the specified consistency rules) (default: **500**). The Rule Cache MRU Size defines the size of the most recently used cache in the storage layer that holds the objects that are not affected by the garbage collector. If the Rule Cache MRU Size is too low, the garbage collector removes the objects and they must subsequently be restored from LDAP when they are accessed the next time, which slows down the agent significantly. We recommend that you set this value to twice the number of rules that are read (the corresponding operations are also cached).

### A.10.14.3. Event-based Maintenance Workflows - Event Attributes

This tab is displayed with the main activity of an event-based maintenance workflow. The available fields depend on the workflow; that is, on the object type the workflow manages.

#### A.10.14.3.1. Event-based User Resolution Workflow

Depending on the changed user attributes, the workflow assigns privileges and resolves

the user, only updates the user's accounts or simply ignores the change event.

This tab allows you to define:

- The user attributes that result in an update of the user's accounts when one of them is changed.
- The search base for finding consistency rules.
- The search base for finding provisioning rules.

Attributes That Trigger Account Update



The accounts are always updated when a permission parameter is changed. This list of attributes is only evaluated if no permission parameter is changed.

You can define "include" and "ignore" attributes. If an "include" attribute was modified, the accounts are updated. If only "ignore" attributes were modified, the accounts are not updated.

You can provide a list of "include" and "ignore" attributes and you can also provide prefix values for them. As an example: Configuring an "include" prefix "dxrSec" means that whenever an attribute beginning with "dxrSec" is changed, the accounts are updated. The configuration items are evaluated in the following sequence:

- If the changed attribute is in the "include" list, accounts are updated.
- If the attribute is in the "ignore" list, the event is ignored.
- If the attribute name matches an "include" prefix, accounts are updated.
- If the attribute name matches an "ignore" prefix, the event is ignored.
- By default, accounts are updated, which means that if all lists are empty, accounts are always updated.

If one condition matches, the succeeding ones are ignored.

**Search Base for Consistency Rules**

Use this optional field to change the search base below which the workflow searches consistency rules. By default, it searches for them in the Policies subtree of the domain.

**Search Base for Provisioning Rules**

Use this optional field to change the search base below which the workflow searches provisioning rules. Leaving it empty means no execution of any provisioning rules, the workflow will ignore applying any provisioning rules.



Privileges assigned by provisioning rules are continuously recalculated during User Resolution. A privilege is retained only if at least one applicable grant rule exists for that privilege during the current execution. If a privilege was previously assigned by a provisioning rule but no matching grant rule is found in the configured search base, the assignment is

considered no longer justified and may be revoked. Therefore, all provisioning rules that can grant a given privilege should be included within the configured search base to avoid unintended privilege removals.

**Sort Key**

Use this optional field to change the search attribute with which the workflow searches rules. The default attribute is **cn**.

**Sort Ascending**

Use this optional field to change the search order in which the workflow searches rules. The default is ascending **TRUE**.

##### A.10.14.3.2. Event-based Persona Resolution Workflow

Depending on the changed persona attributes, the workflow assigns privileges and resolves the persona, only updates the persona's accounts or simply ignores the change event.

This tab allows you to define:

- The persona attributes that result in an update of the persona‘s accounts when one of them is changed.
- The search base for finding consistency rules.
- The search base for finding provisioning rules.
- The name of the link attribute for the persona's owner. This value must be "owner".

**Attributes That Trigger Persona Update**

If the state changes or an attribute changes that is mastered from the user via the "link attribute for the persona's owner (owner)", the persona is saved. Thus, the persona's state is recalculated, taking into account the user's state changes and changes to the attributes changed at the user that are passed to the persona via the master mechanism.

**Attributes That Trigger Account Update**



The accounts are always updated when a permission parameter is changed. This list of attributes is only evaluated if no permission parameter is changed.

You can define "include" and "ignore" attributes. If an "include" attribute is modified, the accounts are updated. If only "ignore" attributes are modified, the accounts are not updated.

You can provide a list of "include" and "ignore" attributes and you can also provide prefix values for them. As an example: Configuring an "include" prefix "dxrSec" means that whenever an attribute beginning with "dxrSec" is changed, the accounts are updated. The configuration items are evaluated in the following sequence:

- If the changed attribute is in the "include" list, accounts are updated.

- If the attribute is in the "ignore" list, the event is ignored.
- If the attribute name matches an "include" prefix, accounts are updated.
- If the attribute name matches an "ignore" prefix, the event is ignored.
- By default, accounts are updated, which means that if all lists are empty, accounts are always updated.

If one condition matches, the succeeding ones are ignored.

**Search Base for Consistency Rules**

Use this optional field to change the search base below which the workflow searches consistency rules. By default, it searches for them under the Personas subfolder of the Policies subtree of the domain.

**Search Base for Provisioning Rules**

Use this optional field to change the search base below which the workflow searches provisioning rules. By default, it searches for them under the Personas subfolder of the Policies subtree of the domain. Leaving it empty means no execution of any provisioning rules, the workflow will ignore applying any provisioning rules.

#### A.10.14.3.3. Event-Based Functional User Resolution Workflow

Depending on the changed functional user attributes, the workflow assigns privileges and resolves the functional user, only updates the functional user's accounts or simply ignores the change event.

This tab allows you to define:

- The functional user attributes that result in an update of the functional user's accounts when one of them is changed.
- The search base for finding consistency rules.
- The search base for finding provisioning rules.
- The name of the link attribute for the functional user's sponsor. This value must be "dxrSponsor".

**Attributes That Trigger Functional User Update**

If the state changes or an attribute changes that is mastered from the user via the "link attribute for the functional user's sponsor (dxrSponsor)", the functional user is saved. Thus, the functional user's state is recalculated, and attributes changed at the user that are passed to the functional user via the master mechanism are updated.

**Attributes That Trigger Account Update**



The accounts are always updated when a permission parameter is changed. This list of attributes is only evaluated if no permission parameter is changed.

You can define "include" and "ignore" attributes. If an "include" attribute is modified, the

accounts are updated. If only "ignore" attributes are modified, the accounts are not updated.

You can provide a list of "include" and "ignore" attributes and you can also provide prefix values for them. As an example: Configuring an "include" prefix "dxrSec" means that whenever an attribute beginning with "dxrSec" is changed, the accounts are updated. The configuration items are evaluated in the following sequence:

- If the changed attribute is in the "include" list, accounts are updated.
- If the attribute is in the "ignore" list, the event is ignored.
- If the attribute name matches an "include" prefix, accounts are updated.
- If the attribute name matches an "ignore" prefix, the event is ignored.
- By default, accounts are updated, which means that if all lists are empty, accounts are always updated.

If one condition matches, the succeeding ones are ignored.

**Search Base for Consistency Rules**

Use this optional field to change the search base below which the workflow searches consistency rules. By default, it searches for them under the Functional Users subfolder of the Policies subtree of the domain.

**Search Base for Provisioning Rules**

Use this optional field to change the search base below which the workflow searches provisioning rules. By default, it searches for them under the Functional Users subfolder of the Policies subtree of the domain. Leaving it empty means no execution of any provisioning rules, the workflow will ignore applying any provisioning rules.

#### A.10.14.3.4. Maintenance Workflows for Business Objects

**Attributes for User Update**

Among other tasks, these workflows update associated users. Some attributes of users are mastered by the business objects to which they are linked. If one of these attributes is modified at the business object, the corresponding maintenance workflow updates these attributes at all associated users.

The field "Attributes for User Update" contains the comma-separated list of these attributes.

**Search Base for Consistency Rules**

Use this optional field to change the search base below which the workflow searches consistency rules. By default, it searches for them in the Policies subtree of the domain.

#### A.10.14.3.5. Maintenance Workflows for Accounts

**Attributes for finding User**

Among other tasks, this workflow tries to find the associated user for an account. It

searches for users whose attributes match those of the account. Use **Attributes for finding User** to configure a sequence of attribute lists.

The processing is similar to finding a joined entry in provisioning workflows: the workflow applies the sequence of attribute lists sequentially until it finds exactly one user. For a search, it takes all attributes from the list of one line and generates a filter to find a user where these attributes match exactly those of the given account. If the search result contains exactly one entry, this is considered to be the associated user. Otherwise, the workflow takes the next list of attributes.

**Search Base for Consistency Rules**

Use this optional field to change the search base below which the workflow searches consistency rules. By default, it searches for consistency rules in the Policies subtree of the domain.

**Search Base for Validation Rules**

Use this optional field to change the search base below which the workflow searches validation rules. By default, it searches for them in the Policies subtree of the domain.

#### A.10.14.3.6. Generic Maintenance Workflows

The generic event-based maintenance workflow can be used for objects of any type. If it receives an event, it

- Executes consistency rules for the object
- Applies the user hook (if configured).

Use this tab to configure the following items:

**Search Base for Consistency Rules**

Use this optional field to change the search base below which the workflow searches consistency rules. By default, it searches for consistency rules in the Policies subtree of the domain.

### A.10.14.4. Event-based Maintenance - User Hook

This tab is displayed with the main activity of each event-based maintenance workflow. Use it to set a user hook and its configuration options. Available parameters are:

**User Hook Classname**

the full Java class name of the user hook class.

**Options**

a table of name / value pairs:

**Property Name**

the name of a configuration option that is evaluated by the user hook.

**Value**

the appropriate value.

See the chapter "Customizing Event-based Maintenance Workflows" in the *DirX Identity Application Development Guide* for information on how a user hook reads these options.

#### A.10.14.5. Joint Backup - Parameters

Use this tab to specify the parameters used for a joint backup of the server content.The Java-based Server triggers the backup to the other components.Available parameters are:

**Message Server**

Leave these fields empty.As of DirX Identity V8.3, each Java-based Server hosts an Apache ActiveMQ Message Broker.Its repository files are no longer backed up.

**Connectivity Configuration**

**Backup Path**

the path and file name where the directory server backup file is stored.The path must exist and should not be below the DirX installation tree.

**Windows Platform**

whether (checked) or not (unchecked) the Connectivity configuration directory server runs on a Windows platform.

**Provisioning Configuration**

**Backup Path**

the path and file name where the directory server backup file is stored. This field is only necessary if you have Connectivity and Provisioning separated onto two DirX LDAP servers. The path must exist and should not be below the DirX installation tree.

**Windows Platform**

whether (checked) or not (unchecked) the Provisioning configuration directory server runs on a Windows platform.

**Related Topics**

Joint Backup - Post Operation

#### A.10.14.6. Joint Backup - Post Operation

Use this tab to specify the parameters used for post operation configuration of the joint backup. Available parameters are:

**Target Backup System Path**

the path and file name where the backup files are stored after the backup operation.This will only work if the IdS-J server can access all paths (the Connectivity and the Provisioning backup paths specified in Joint Backup - Parameters).You must use a different path from the paths specified for Connectivity and Provisioning.

**Windows Platform**

whether (checked) or not (unchecked) the messaging service runs on a Windows platform.

**Userhook Class Name**

(not used).

**Related Topics**

Joint Backup - Parameters

# A.11. Schedules

A folder for the schedule configuration objects in the configuration database.

**Name**

the name of the folder.

**Description**

descriptive text for this object.

The icon of this folder turns to red when any of the contained schedules is active and when scheduling is not disabled.

If you use the DirX Identity Manager menu entry **Disable Scheduling**, all schedulers on all servers stop scheduling immediately.You can enable scheduling again with the menu entry **Enable Scheduling**.See the DirX Identity Manager online help for details.



that disable scheduling does not interrupt or abort running workflows. Only the new start of workflows is prevented.

For Java-based workflows, note that the Java-based Server scheduler looks for schedules only in a domain-specific subfolder (with the name of the domain).

Within a property page, the content of this folder is shown as a pop-up list of a combo box:

**Schedule**

the schedule object currently used by the object of which the properties are shown. Use the arrow button to pop up the list of all available schedule objects in the schedule folder. Use the properties button to display the properties of the currently selected schedule object.

**Related Topic**

Schedule
Ranges
Target System Cluster

Workflow Design Rules

### A.11.1. Schedule

A schedule links to a workflow and determines when and how often it runs.The schedule configuration object describes a particular schedule.Use the schedule configuration object to define a schedule for a workflow and associate it with the workflow.You can set up schedules for Tcl-based and Java-based workflows.

A schedule configuration object defines the time controls for the workflow.For example, the workflow starts first at midnight on 21 December 2000 and is to run every 24 hours (the interval).You can define an expiration time after which this workflow should not run again.

A schedule configuration object can also specify a range after the scheduled start time where it's okay to start the workflow (the deviation).Use this workflow run range to account for possible system or network failures that can postpone the workflow's execution.

Additionally you can define, for Tcl-based workflows, a retry interval for each schedule.This feature triggers automatic restart when a workflow ran into a temporary error situation (for example the network was temporarily not available).Note that this feature is not available for real-time workflows.

For example, suppose you have a workflow that generally runs for three hours.You schedule it to start at midnight (the start time) plus four hours (the deviation), after which it cannot run until midnight the next day.If a system crash or network failure occurs and postpones your workflow run past 4 AM, it will not run, and will therefore not interfere with interactive accesses in the morning when people start to work again.

You can switch off scheduling completely with the option **Disable/Enable Scheduling** at the Schedules configuration object in the DirX Identity Manager Expert View. See the chapter on the DirX Identity Manager in the *DirX Identity User Interfaces Guide* for more information about the Expert View.

For Java-based workflows, note that the Java-based Server scheduler looks for schedules only in a domain-specific subfolder (with the name of the domain).

Use this tab to set the properties of a schedule. The available parameters are:

**Name**
: the name of the schedule.

**Description**
: a description of the schedule.

**Version**
: the version number of the schedule object.

**Active**
: whether the schedule is active (checked) or not (unchecked). Only active schedules are executed by the Scheduler.

**Workflow**

the workflow associated with the schedule. To display its properties, click the Properties button  on the right.

**Start Time**

the time and date at which the schedule starts. This is the time at which the workflow is started the first time.

**Time Interval**

the interval at which the schedule will run the workflow.

**Expiration Time**

the expiration time and date, if any, for the schedule. This is the last time when the workflow will be executed. If no expiration time is defined, the schedule is active forever.

**Deviation**

the maximum allowed deviation for the schedule. This is a plus range around the **Start Time**. It allows you to suppress workflow runs that would start too late (for example, because of a network problem).

**Retry Interval**

the frequency of retries when the workflow encounters an error condition. The workflow is repeated after this time as long as the deviation time has not passed. This feature allows overcoming sporadic errors (for example, network errors).



Warnings do not enforce a retry.



This property is not evaluated for Java-based workflows. Instead, the "wait before retry" and "retry limit" properties at the workflow activities are used.

Active schedules show a red icon to indicate this state. Nevertheless, you can deactivate scheduling centrally (see Schedules). See the topic "Workflow Design Rules" for information about setting scheduling parameters.



All parameters that contain a date are stored in GMT format, which means that a workflow runs at a fixed world time. However, the display of these parameters is in local time. This setup allows for consistent handling of workflow starts in a worldwide distributed meta directory scenario.



Working in a country that switches between summer and winter time (daylight savings time) may require adjusting the schedules so that they start in relation to fixed local time. We recommend that you:

- Set all start and end dates in the winter time range.
- Use the script **shiftSchedules.tcl** that reads all schedule start and end dates and adjusts them accordingly (+ 1 h or - 1 h). See the section Daylight savings time for more information.

- Use the operating system scheduler to start this script regularly when winter or summer time begins.

**Related Topics**

Ranges
Target System Cluster

Schedules
Workflows

Operation

## A.11.2. Ranges

Schedules are used to start the execution of a workflow at a predefined date and time.Use this tab to set the ranges properties of a schedule.The available fields are:

**Ranges**

the daily or weekly restrictions defined for workflow runs.Define ranges graphically in half-hour steps.White areas indicate disabled, gray areas indicate enabled.You can choose either **GMT** or **Local** time.Select a range with the left mouse button and then select **Enable** or **Disable** from the context menu. **Enable All** or **Disable All** allows activating or deactivating the whole range.

Active schedules show a red icon to indicate this state.Nevertheless, you can deactivate scheduling centrally (see the topic "Schedules").See the topic "Workflow Design Rules" for information about setting scheduling parameters.



All parameters that contain a date are stored in GMT format, which means that a workflow runs at a fixed world time.However, the display of these parameters is in local time.This setup allows for consistent handling of workflow starts in a worldwide distributed meta directory scenario.



Working in a country that switches between summer and winter time may require adjusting the schedules so that they start in relation to fixed local time.We recommend that you:

- Set all start and end dates in the winter time range.
- Write a small script that reads all schedule start and end dates and adjusts them accordingly (+ 1 h or - 1 h).
- Use the operating system scheduler to start this script regularly when winter or summer time begins.

**Related Topics**

Schedule
Target System Cluster
Schedules

Workflows

### A.11.3. Target System Cluster

Use this tab to define a set of target systems that this schedule handles (this feature is only available for real-time workflows).Available parameters are:

**Search base**

the search base for the LDAP search.The default is the cluster container for the relevant target systems.

**Filter**

the LDAP filter condition.This field allows you to separate sets of workflow runs.

See the section "Using Cluster Workflows" to understand the concept in detail.

**Related Topics**

Schedule
Ranges

## A.12. Scenarios

### A.12.1. Workflow Line

Use this tab to set the properties of a workflow line.Available properties are:

**Name**

the name of the workflow line.

**Version**

the version number of the workflow line.

**Source**

the source connected directory for the workflow line.To display its properties, click the Properties button  on the right.

**Target**

the target connected directory for the workflow line.To display its properties, click the Properties button on the right.

**Workflow**

the workflows represented by the workflow line. To display the properties of one of these objects, click the Properties button on the right.

**Related Topics**

Connected Directory
Connected Directory Icon

Scenario
Workflows

## A.12.2. Connected Directory Icon

A connected directory icon is the graphical representation of a connected directory in a meta directory scenario.

Use this tab to set the properties of the connected directory icon by hand.The available properties are:

**Name**

the name of the connected directory icon.

**Version**

the version number of the connected directory icon.

**X**

the horizontal (X) position of the icon in the scenario map.

**Y**

the vertical (Y) position of the icon in the scenario map.

**Moveable**

whether (checked) or not (unchecked) the connected directory can be moved over the map.

**Connected Directory**

the connected directory that the icon represents.To display its properties, click the Properties button  on the right.

**Related Topics**

Connected Directory
Scenario

## A.12.3. Scenario

A synchronization scenario is a specific set of connected directories and synchronization workflows that comprises your complete meta directory environment.For example, the elements of a scenario can include:

- A human resources database that is the master for names and addresses
- A PBX database that is the master for telephone numbers
- Windows Active Directory domains that are the masters of accounts and email addresses
- The Identity store that is the central manager for this data and is also the central repository for user-related data (white book)

Workflows connect and synchronize all of these directories.

You can set up different scenarios in one configuration database either to separate independent parts of your directory environment or to distinguish between a production environment and a test environment.

The scenario configuration object describes a synchronization scenario. Each scenario configuration object describes the configuration information for a particular synchronization scenario. Scenario configuration objects are associated with connected directory objects and workflow objects.

A scenario is usually displayed in the Global View as a map in the with "tin" icons for connected directories and lines between them representing synchronizations workflows. Scenario properties are usually set from the Global View. Use this tab to set the properties of a scenario by hand.

**Name**
The name of the scenario.

**Description**
descriptive text for this scenario.

**Version**
the version number of the scenario.

**Icons**
lists the connected directory icons contained in the scenario. To display the properties of a selected icon, click the **Properties** button to the right.

**Lines**
lists the workflow lines contained in the scenario. To display the properties of a selected workflow line, click the **Properties** button to the right.

The following properties can also be accessed by invoking the "Properties..." item of the pop-up menu displayed when you right-click on the Global View's map:

**Use Grid**
whether the grid is on (checked) or off (unchecked).

**Grid-X**
the grid cell width.

**Grid-Y**
the grid cell height.

**Map**
the pathname of the background image displayed in the scenario map. This value can be either a usual file path or a path that is relative to the class path (the value of the CLASSPATH environment variable).

**Related Topics**

Connected Directory Icon
Workflow Line

### A.12.4. Scenarios

A folder for the scenario configuration objects in the configuration database. Use this tab to name the folder.

**Name**

the name of the folder.

**Description**

a description of the folder content.



Scenarios can be structured in folders. This structure is also shown in the tree pane of the Connectivity Global View.

**Related Topic**

Scenario

## A.13. Tcl-based Workflows

### A.13.1. Tcl-based Activities

Use this tab to view the activities associated with a Tcl-based workflow.

**Activities**

the associated activities, displayed in a table.

To display the properties of an activity, click it and then click .

To insert a new activity, click  on the right side of the table.

To delete an activity, click it and then click .

To display the distinguished names of the associated activities in text format, click  on the right side of the table.

**Related Topics**

Tcl-based Activity
Job
Tcl-based Workflow

## A.13.2. Tcl-based Activity

An activity is an instance of a job and a single step in a Tcl-based workflow.An activity configuration object describes the configuration information for a particular step in the workflow and maintains all data needed for execution within the running workflow.Use the activity configuration object to describe each activity and the required sequence of activities that you want to establish for a workflow in your Identity environment.

The configuration data associated with an activity includes:

- The activity's name, description and version number
- The job or workflow to which the activity is linked
- The activity's position in the workflow's control flow

Use this tab to enter the properties of an activity object.The items shown in this tab are:

**Name**

the name of the activity.

**Description**

the description of the activity.

**Version**

the version number of the activity.

**Run Object**

the object to be run that is associated with the activity. Run objects can be jobs or workflows. To display the properties of a run object, click it and then click .

**C++-based Server**

the C++-based Server associated with the activity. To display its properties, click it and then click .

**Ignore Errors**

whether (checked) or not (unchecked) this activity is prevented from stopping the workflow if the activity encounters an error condition. This flag is especially useful for workflow run objects. Nested workflows should normally not be interrupted if one of their child workflows is not successful.



Do not use this flag for jobs that use delta handling. This could result in serious data loss! (For details on delta handling see the *Application Development Guide* → Understanding the Default Application Workflow Technology → Understanding Tcl-based Workflows → Delta Handling.)

**Disable Statistics**

whether (checked) or not (unchecked) to prevent propagation of the statistics information from the activity to the workflow object.

**Is Start Activity**

whether (checked) or not (unchecked) the activity is the starting activity.

**Predecessor**

the predecessor activity for this activity. To display the properties of an activity, click it and then click . Note that the start activity cannot have a predecessor.

**Is End Activity**

whether (checked) or not (unchecked) the activity is the terminating activity.



You can set this flag to abort a workflow at this point for test purposes.If this activity is referenced by another activity as Predecessor, a warning is created during the run of this workflow (only visible in the event viewer / logging).

**Anchor**

the text value that helps to select the correct activity during reference resolution.See the chapter "Customizing Object References" in the *DirX Identity Customization Guide* for details.

**Related Topics**

Job
Tcl-based Workflow

## A.13.3. Tcl-based Channel

Channels link jobs to connected directories and describe bulk data input from and output to the connected directory.A channel configuration object describes the configuration information for a particular channel.

Channels are always located under a connected directory object.Thus they can either reside in the connected directories folder or in the jobs folder as sub-objects of connected directory objects.A channel is the connector between a connected directory and a job and therefore holds all data necessary for a particular download or upload operation of bulk data.

The configuration information associated with a channel includes:

- The channel's name, description and version number
- The **connected directory** to which the channel is linked
- The channel's **role**.This role can be used for different purposes; for example, the meta controller uses it to define the handles in the Tcl scripts.In most simple cases, a single input and output channel are sufficient.However, more complex cases can exist where input from different channels is needed to build an output entry.In this case, the role property can be used to distinguish between the different channels.

Depending on the type of connected directory with which the channel is associated, it can also store information about:

- **Export parameters**, also called search parameters (the vertical or "y" axis selection criteria) to define the export operation from a connected directory. Export parameters define the set of entries retrieved from a source connected directory (for example, a base object or a filter).
- The set of attributes to be synchronized (the horizontal or "x" axis selection criteria), which is also called "**selected attributes**". A channel's selected attributes list is used to define appropriate mapping procedures between one or more input attributes and one or more output attributes.
- **Import parameters** to define how objects are handled during import operations into a connected directory. Import parameters define the modes for entry input in the target connected directory. They also define the rules for the treatment of new objects (is it permitted to add new entries?), for renaming objects (is it permitted to rename entries?) or for the deletion of objects (perhaps some entries must not be removed; for example, the login account for unattended operation must be kept in all cases).

The channel configuration object can have additional properties depending on the type of connected directory. Channels are bound to a specific instance of a connected directory and therefore inherit the properties of the corresponding type.

Use this tab to assign general properties to a channel configuration object. The properties shown in this tab are:

**Name**
: the name of the channel.

**Description**
: the description of the channel.

**Version**
: the version number of the channel.

**Connected Directory**
: the connected directory associated with the channel. To display its properties, click the Properties button on the right.

**Channel Type**
: the channel type. Each connected directory can have different types of channels, which is reflected by this property. If you choose another type of channel, the display behavior of this channel object will change after restarting the DirX Identity Manager or after performing **Reload Object Descriptors** in the DirX Identity Manager Expert View.

**Role Name**
: the role name of the channel. For an LDAP directory, the default is "ldap". For a file, the default is "file". This parameter is used in mapping Tcl files to define the handles for the meta controller. Be sure to set correct references in the Tcl files or adapt the Tcl files by hand if you change this parameter.

**Anchor**
: the unique easily readable identifier for the channel to be used in references. See the

chapter "Customizing Object References" in the *DirX Identity Customization Guide* for details.

**Bind Profile**

the bind profile in the connected directory to be used for the bind operation. (Channels define how to access the connected directory.)

**Dereference Alias**

whether aliases are always de-referenced (select **Always**) or never de-referenced (select **Never**). LDAP channels can use this parameter.

**Referral Handling**

whether or not the LDAP library is to follow the referrals provided by the LDAP server. LDAP channels can use this parameter. Possible values are:

**Follow Referral**

the LDAP library evaluates referrals returned by the LDAP server. If it is unable to evaluate the referral, the referral is returned in the search result and the Tcl workflow terminates with errors due to an incomplete search result.

**Don't Follow Referral**

the LDAP library ignores referrals returned by the LDAP server. The Tcl workflow doesn't terminate due to an incomplete search result. It is the responsibility of the user /administrator to decide whether the behavior is acceptable or not.

There may be additional channel properties associated with a specific channel, depending on the respective agent and connected directory type.

File-type connected directories (for example, CSV or LDIF) display the following additional property:

**Internal**

whether (checked) or not (unchecked) the channel is job-internal.A job-internal channel is not used outside the job and is not shown in the list of usable input or output channels for the job.

**Related Topics**

Connected Directory
Export Properties
Selected Attributes
Specific Attributes

## A.13.4. Data Flow Viewer

The Data Flow Viewer allows you to display and maintain the data flow in and out of a connected directory for Tcl-based workflows.It works on existing workflows, especially the selected attributes of a channel.You access the Data Flow Viewer from a scenario in the Global View by right-clicking on a connected directory icon and then selecting **Data Flow...**

The Data Flow Viewer window comprises four sub-windows:

**Available attributes**

the attributes of the attribute configuration of the connected directory.Click an attribute in this list to select it.Use the SHIFT key to select a range and the CTRL key to add or remove single attributes to or from the selection.Use CTRL-A to select all attributes.

**Display Data Flow for**

the attributes for which the data flow is displayed.Use the up arrow  to add attributes from the **Available attributes** list, use the down arrow  to remove attributes from this list.

**(attribute columns)**

a column for each selected attribute.The column header displays the name of the attribute.The rest of the column contains direction indicators that show the data flow for this attribute.These indicators are:



the attribute is **exported** from the connected directory listed in the **Connected Directories** column and is imported into the connected directory where the Data Flow Viewer was started.



the attribute is exported from the connected directory where the Data Flow Viewer was started and **imported** into the connected directory listed in the **Connected Directories** column.

**Connected Directories**

the target system (connected directory) to which the attribute is imported or from which it is exported.

Envision the connected directory where the Data Flow Viewer was started on the left side and all the other connected directories on the right side. Then the arrows show the direction of the data flow. The following figure shows an example for the attributes c and l.

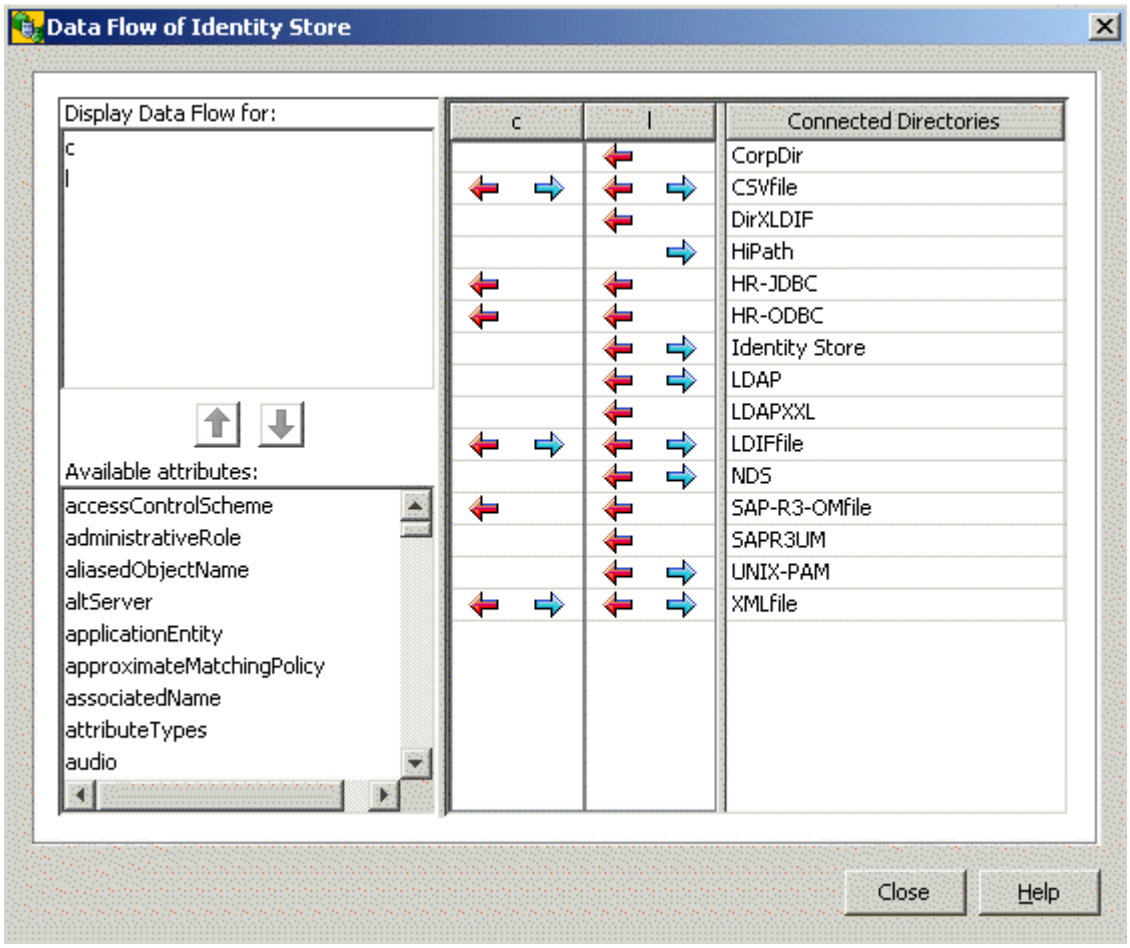



*Figure 30. Data Flow Viewer*

The arrows in the second column in the figure show the data flow for the c attribute. This attribute is exported to the CSVfile, LDIFfile and XMLfile connected directories and imported from these directories and also from the SAP-R3-OMfile.

To analyze the details of an arrow, right-click it and then select **Modify Data Flow**. You can also just double-click the arrow. See the Data Flow Editor for an explanation of the resulting display.

- The Data Flow Viewer does not recognize additional selected attributes added by If Tcl scripts.
- Depending on the complexity of the scenario, the recalculation of the data flow after starting the Data Flow Viewer can take a few seconds.Adding or removing attributes in a loaded Data Flow Viewer is much faster.
- You can open several Data Flow Viewers at the same time.

**Related Topics**

Data Flow Editor

## A.13.5. Data Flow Editor

The Data Flow Editor is opened from the Data Flow Viewer when you right-click on an arrow in a selected attribute column and then select **Modify Data Flow**.The Data Flow Editor displays all workflows and activities that caused the import or export arrows for the selected attribute displayed in the Viewer.It also shows the Selected Attributes tabs of the related channels and the mapping between them.You can edit all of this information.After you close the Editor window, the Data Flow Viewer window is updated with your changes.The following figure shows the Data Flow Editor dialog.

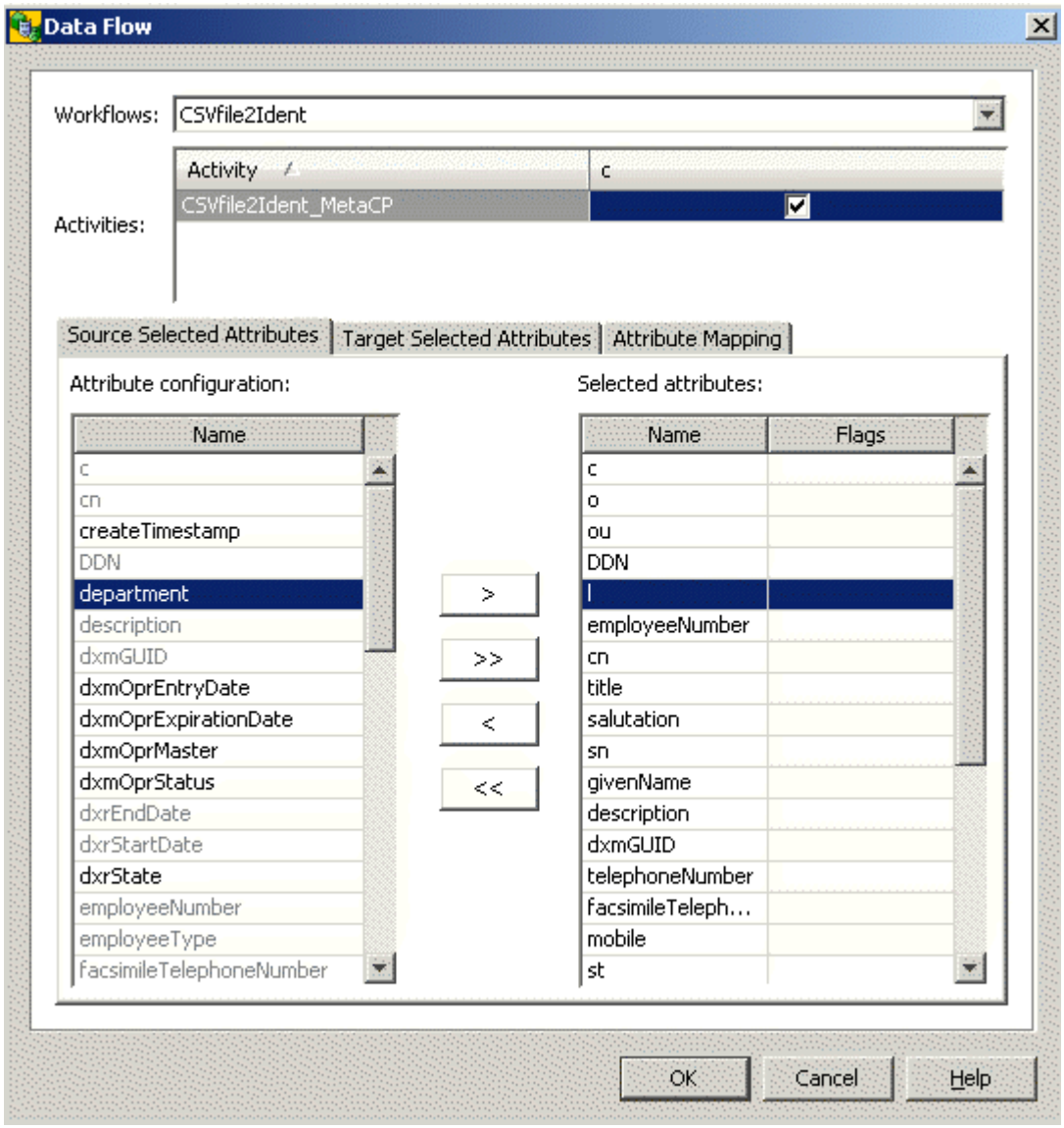



*Figure 31. Data Flow Editor*

The Data Flow Editor consists of the following items:

**Workflows**

the workflows that handle the requested attribute.Select one from the list.

**Activities**

the activities of the selected workflow. All activities that contain the requested attribute in the source selected attributes are marked in the right-hand column (the column

header shows the name of the requested attribute).

**Source Selected Attributes**

the Selected Attributes tab of the input channel of the selected activity. It contains the requested attribute.

**Target Selected Attributes**

the Selected Attributes tab of the input channel of the selected activity. This table must not contain the requested attribute.

**Attribute Mapping**

the mapping of the selected activity. This table must not contain the requested attribute, but it often does. Note that the mapping can also be done in the pre-, post- and post-join mapping parts (the buttons below the mapping table).

You can edit the information in the Source and Target Selected Attributes tabs and also in the Attribute Mapping tab. Changing the Source Selected Attributes results in an updated Data Flow Viewer window when you click **OK**. Cancel aborts the editing procedure.



- The requested attribute should not appear in the Attribute Mapping tab if it can be used for other purposes (for example, for a search operation).
- The Data Flow Viewer does not recognize selected attributes added by Tcl scripts.

**Related Topics**

Data Flow Viewer
Selected Attributes

"Using the Mapping Editor" in the *DirX Identity User Interfaces Guide*

## A.13.6. Entry Handling

Entry-handling parameters specified in the output channel definitions of Tcl-based workflows define the entry-handling behavior at the target connected directory side.The parameters displayed depend on the workflow.Use this tab to specify these parameters.

Use the Add Properties section of this tab to control how the addition of entries is handled:

**Add Entries**

whether or not entry additions are allowed and whether or not notification is performed.Possible values are:

**NONE** - No addition

**ADD** - Addition only

**NTF** - Notification only

**ADDNTF** - Addition and notification

**Superior Info**

information for building higher-level nodes in the target system if not already present. Use the superior info editor to customize this information (click the button to the right of the text field to enter the editor).

**Source Add Marking Attribute**

the attribute type that is used as a status attribute in the source connected directory.

**Source Add Marking Value**

the attribute value that indicates an addition in the source connected directory.

**Target Add Marking Attribute**

the attribute type that is used as status attribute in the target connected directory and that must be set if the Source Add Marking Attribute has the Source Add Marking Value.

**Target Add Marking Value**

the attribute value that indicates an addition (for example, NEW) in the target connected directory.

Use the Modification Properties section of this tab to control how the modification of entries is handled:

**Modify Entries**

whether or not modifications are allowed and whether or not notification is performed. Possible values are:

**NONE** - No modification

**MOD** - Modification only

**NTF** - Notification only

**MODNTF** - Modification and notification

**Modify Marking Attribute**

the attribute type that is used as a status attribute in the target connected directory when the entry is modified.

**Modify Marking Value**

the attribute value that indicates a modification (for example, MOD) in the target connected directory.

**Rename Entries**

whether a modification of the distinguished name (DN) is allowed (True) or not (False). Possible values are:

**FALSE** - move DN not allowed

**TRUE** - move DN allowed

Use the Deletion Properties section of this tab to control how the deletion of entries is

handled:

**Delete Entries**

whether or not deletions are allowed and whether or not notification is performed. Possible values are:

**NONE** - No deletion

**DEL** - Deletion only

**NTF** - Notification only

**DELNTF** - Deletion and notification

**Deletion Mode**

how the deletion is performed. Possible values are:

**PHYSICAL** = the entry is physically removed

**MARK** = the entry is marked

**MOVE** = the entry is moved to a tombstone area

**USER** = a user hook defines the deletion mechanism

**Source Delete Marking Attribute**

the attribute type that is used as a status attribute in the source connected directory.

**Source Delete Marking Value**

the attribute value that indicates a deletion in the source connected directory.

**Target Delete Marking Attribute**

the attribute type that is used as a status attribute (if Deletion Mode = MARK) in the target connected directory and which must be set if the Source Delete Marking Attribute has the Source Delete Marking Value.

**Target Delete Marking Value**

the attribute value that indicates a deletion (if Deletion Mode = MARK), for example DEL.

**Keep Objects**

the list of objects that cannot be deleted.



DirX Identity automatically ensures that it does not delete the object that was used for the directory bind operation. As a result, you don't need any additional protection for this object.

For the HiPath workflow, the following properties are displayed:

**Keep unmastered attributes**

whether (checked) or not (unchecked) values for attributes in the Multi Master Attribute List are to be kept if they are not managed by an attribute master (for example, manually entered by an administrator).

**Multi Master Attribute List**

the list of attributes that are mastered by multiple entries from possibly multiple connected directories that are attribute masters for these attributes.Using the dxmOprOriginator attribute, the HDMS workflow is able to update exactly those attribute values related to a particular HDMS entry.By default, these attributes are **telephoneNumber** and **facsimileTelephoneNumber**.This enables the HDMS workflow to link multiple telephone numbers and facsimileTelephoneNumbers of a person to one single meta directory entry.

**Related Topics**

Tcl-based Channel
Import Properties

## A.13.7. Export Properties

Export properties are necessary for setting up the specific options for exporting data from a connected directory.Usually these parameters describe the subset of entries and the attribute set to be downloaded.

Use this tab to specify these export properties.The parameters displayed depend on the type of connected directory and the type of agent.

For the ADS and Exchange connected directories, the following parameters are displayed:

**Search base**

the base node within the connected directory from which to export entries.



This field can contain references.See the chapter "Customizing Object References" in the *DirX Identity Customization Guide* for more information.

**Search scope**

the scope of the search operation. Possible values are:

**0-BASE OBJECT**

check that the Base Object matches the search criteria but nothing else.

**1-ONE LEVEL**

search for results in the given Base Object node and one level deeper.

**2-SUBTREE**

find results in the entire subtree below the given Base Object.

**Search filter**

the collection of entries to locate and export. This value must be an LDAP-conformant expression.



This field can contain references. See the chapter "Customizing Object

> References" in the *DirX Identity Customization Guide* for more information.

**Page size**

the page size for the search results. This parameter controls how the connected directory will return search results for a single search operation. If the value is **0**, the entire search result is processed before returning to the agent. If the given number is *n*, the result is split into pages with *n* entries at the maximum.

**Paged time limit**

the length of time the connected directory is allowed to search for a single page.

**Asynchronous search**

whether synchronous (unchecked) or asynchronous (checked) search operations are performed.

**Cache results**

whether (checked) or not (unchecked) the connected directory agent caches search results in its local memory.

**Time Limit**

whether the connected directory imposes a time limit for search results to be returned from connected directory server.

**Select Attributes**

whether or not the ADSAgent retrieves all entry attributes with a value or only those attributes set to 1 in the Attributes section.

**Multi-value separator**

the value to be used to separate the individual attribute values of a multi-valued attribute.

For the HDMS connected directory, the following parameters can appear in addition to the items in the **Identity Store** section:

**Joinback expression**

The joinback expression that enables the HDMS HiPath workflow to detect *n*:1 relationships between Identity store entries and HDMS entries. This parameter and the HDMS join expression define the best-guess-match policy that the HDMS HiPath workflow is to use to join Identity store and HDMS entries. Each attribute in the joinback expression must correspond one-to-one with an attribute in the HDMS join expression. For example:
Joinback Expression: surname and givenname
HDMS Join Expression: name and christianname

In this example, HDMSAgent is to match Identity store entries with HDMS entries using a combination of surname and given name. The Identity store sn (surname) attribute maps to the HDMS attribute “name” and the Identity store gn (given name) attribute maps to the HDMS attribute “christianname”.

We strongly recommend using attribute combinations that make a connected directory unique (similar to our default setting). The workflow is able to detect ambiguities, but these ambiguities must be resolved manually by administrators.

For the Identity Store connected directory, the following parameters are displayed:

**Base Object**
the node at which the search operation starts for entries to be exported.

**Subset**
the scope of the search operation. Supply one of the following values:

**BASE OBJECT**
check that the Base Object matches the search criteria but nothing else.

**ONE LEVEL**
search for results in the given Base Object node and one level deeper.

**SUBTREE**
find results in the entire subtree below the given Base Object.

**Object Class Collection**
the set of object classes to use to create new objects. These sets are defined in the Operational Attributes tab of the connected directory (attribute Object Classes).

**Search Filter**
the matching criteria. This value can be any logical expression in LDAP notation. Use "&" for "AND", "|" for "OR", and "!" for "NOT", as well as "=" for equality check, "-=" for phonetic equality check, and "⇐" for less, ">=" for greater.
When delta mode is on, this filter is combined to &(*search_filter*)(*delta_filter*) where *delta_filter* selects all entries that have been created or modified since the last successful run.
Note: This field can contain references. See the chapter "Customizing Object References" in the *DirX Identity Customization Guide* for more information.

**OR Filter**
an additional OR condition (for possible expression,s see **Search Filter** ). The complete filter to be used is composed of:
|(&(*search_filter*)(*delta_filter*))(*or_filter*)
The Delta Filter is used to retrieve delta data if Delta Mode is checked. The OR filter can be used if the delta filter and the search filter does not retrieve all necessary data. An additional set of entries can be searched this way.
Note: This field can contain references. See the chapter "Customizing Object References" in the *DirX Identity Customization Guide* for more information.

**Sorted list**
whether (checked) or not (unchecked) the resulting entry list should be sorted.

**Sort key**

the sort key to use. Select an attribute name from the combo box that is used for sorting.

**Sort order**

whether the results are sorted in **ASCENDING** or **DESCENDING** order.

**Read DN only**

enables (**True**) or disables (**False**) the optimization for lower memory consumption. Note: This flag was previously named **Perform Read**.

**False**

searches complete entries with all attributes during the first search. No additional search operations are necessary later on. This requires a lot of memory but results in maximum speed.

**True**

searches only DNs during the first search and later on all other attributes for each entry by an additional search operation. This lowers memory consumption enormously but decreases performance by more than 50 %.

**Paged Read**

whether (checked) or not (unchecked) the script works in paged mode if supported by the LDAP server.

**Page Size**

the size of the pages. Default is 1000 bytes.

For the IBM Notes connected directory, the following parameters are displayed:

**(Addr)ess Book**

the name of the Notes address book from which entries are to be exported.

**Form Name**

the Notes document type to be extracted from the Notes address book.

**Search Documents**

whether or not the Notes agent searches for and exports specific entries ("documents", in Notes terminology) of the document type specified in the **Form Name** field of the channel's property tab.

**Search Item Name**

the attribute within an entry to search for.

**Search Item Value**

the value to search for (case-exact match), given an attribute name to search for that is specified in the **Search Item Name** field.

**Update Export File**

(for delta exports only) whether or not the full export data file created by the agent can

be updated.

**Copy Deleted Addresses in Modification File**

whether or not the agent retrieves the contents of entries of the document type selected with the **Form Name** field of the channel's property tab sheet that have been deleted since a full export data file was last generated.

**File for Modified Addresses**

the path name of the file to which the agent is to write modified (and deleted) entries during a delta export operation.

**First Delta is Full**

whether NTAgent also writes all entries into the "modify" file.

**Export All Items**

whether all of the attributes ("items", in Notes terminology) of entries of the document type selected with the FormName field are exported, or whether a specified subset of attributes is exported.

**Multi-Value Separator**

the value to be used to separate the individual attribute values of a multi-valued attribute.

**SMTP Host Domain**

whether or not Internet addresses for Person entries exported from a Notes address book are generated.

For the NT connected directory, the following parameters are displayed:

**Data Type**

the type of exported data. Supply one of the following values:

**1-ACCOUNTS**

Account data will be exported.

**2-GLOBAL GROUPS**

Global groups will be exported

**3-LOCAL GROUPS**

Local groups will be exported

**4-ACCOUNTS AND GLOBAL GROUPS**

Accounts and global groups will be exported

**5-ACCOUNTS AND LOCAL GROUPS**

Accounts and local groups will be exported

**6-ALL GROUPS**

Global and local groups will be exported

**7-ACCOUNTS AND ALL GROUPS**

Accounts, local and global groups will be exported

**User Data**

the attributes of the exported data. Supply one of the following values:

**1-STANDARD**

The data are exported with a standard attribute set (*UserName, Comment, FullName, UserID*).

**2-EXTENDED**

The data are exported with the STANDARD attribute set extended by resource and account attributes.

**3-STANDARD WITH DIALIN**

The data are exported with the STANDARD attribute set extended by the dial-in attributes (*DialInPrivlege* and *DialInPhoneNumber*).

**4-EXTENDED WITH DIALIN**

The data exported with the EXTENDED attribute set extended by the dial-in attributes (*DialInPrivilege* and *DialInPhoneNumber*).

**User Group Data**

Selects the exported group data. The field can take one of the following values:

**0-NONE**

Do not export group names of which the account is a member

**1-GLOBAL GROUPS**

Export any global group names of which the account is a member

**2-LOCAL GROUPS**

Export any local group names of which the account is a member

**3-ALL GROUPS**

Export any global and local group names of which the account is a member

**First Delta is Full**

whether NTAgent also writes during the first full export all entries into the “modify” file.

**Multi-Value Separator**

the value to be used to separate the individual attribute values of a multi-valued attribute. It has the same syntax as the Separator field in the export configuration file.

**Date Format**

the format of date values.

**Date Separator**

the separator character of date items.

For the ODBC connected directory, the following parameters are displayed:

**From**

the table or tables in the ODBC database from which agent is to extract entries. If the data will be exported from more than one table, the table names must be separated by a comma.



This is multi line field. Each line must end with '\' as last character!

**Where**

whether or not the agent searches for and exports specific entries ("rows" in ODBC terminology). This must be an expression in SQL syntax.
Note: This is a multi-line field. Each line must end with a backslash (\) as the last character.

**Keys**

the set of attributes that the agent is to use to uniquely identify each entry to be exported from the ODBC database.

**Sort Control**

enables/disables sorting optimization. When the agent creates a reference file for delta export, it sorts the records in the file by ordering the key fields. This ordering then permits fast analysis of changes between the previous state of the database and the present one, and allows the modified information to be selected. The process of sorting and extraction is much faster if the sorting of the reference file information corresponds to the order in which the ODBC database is sorted. The **Sort Control** parameter enables the sorting to be optimized where necessary. In many cases, the sorting will be correct anyway.

**Reference Path**

the path to the directory in which the agent will store delta export reference files.

**Save Attributes**

enables the delta reference information to be related to information in the target database. For example, if an entry is removed from the ODBC database, it may be required to remove the corresponding entry in the target database; the **Save Attributes** parameter is used to specify any additional attributes that can be used to identify the entry in the target database that is to be removed. Otherwise, removal is impossible.

For the SAP EP UM connected directory, the following parameters are displayed:

**Search Parameters**

the search base in the SAP Enterprise Portal.

**Scope**

the search scope. Enter one of the following values:

**BASE OBJECT**

check that the Base Object matches the search criteria, but nothing else.

**ONE LEVEL**
search for results in the given Base Object node and one level deeper.

**SUBTREE**
find results in the entire subtree below the given Base Object.

**Filter Attribute Name**
the attribute name that is to be used as filter property.

**Filter Attribute Value**
the attribute value that is to be used as filter criteria.

See the "Export Configuration File Format" sections in each agent chapter in the *DirX Identity Connectivity Reference* for complete details of each DirX Identity agent's export parameters.

**Related Topic**

Channel

## A.13.8. Import Properties

Import properties are needed to set up the specific options for importing data into a connected directory.Usually, these parameters describe the restrictions during data import.Use this tab to specify these import properties.The actual parameters are determined by the type of connected directory and the type of agent.

For the **ADS** connected directory, the following fields appear (see the ADS agent chapter in the *DirX Identity Connectivity Reference* for details and examples for allowed values of these fields):

**Search Base**
the base within the Active Directory from which to search for matching entries using the search criteria specified in the *ldapFilter* attribute of each entry in the import data file.

**Ignore Object Class**
whether ADSAgent evaluates (checked) or ignores (unchecked) the *objectClass* attribute of entries for which the *modify* LDIF changetype operation has been specified.

**Reject Special Characters**
whether ADSAgent evaluates (checked) the ADsPath attribute of entries in the import data file for special characters or not (unchecked).

**Rejected Characters**
the characters in the import entries' AdsPath attribute for which ADSAgent is to scan; ADSAgent is to reject the entry for import if it contains one of these characters.

**Multi-value Separator**
the value to be used to separate the individual attribute values of a multi-valued attribute.

For the **Exchange** connected directory, following fields appear (see the Exchange agent chapter in the *DirX Identity Connectivity Reference* for details and examples for allowed values of these fields):

**Search Base**

the base within the Exchange Directory from which to search for matching entries using the search criteria specified in the *ldapFilter* attribute of each entry in the import data file.

**Ignore Object Class**

whether ExchangeAgent evaluates (checked) or ignores (unchecked) the *objectClass* attribute of entries for which the *modify* LDIF changetype operation has been specified.

**Reject Special Characters**

whether ExchangeAgent evaluates (checked) the ADsPath attribute of entries in the import data file for special characters or not (unchecked).

**Rejected Characters**

the characters in the import entries' AdsPath attribute for which ExchangeAgent is to search; ExhangeAgent is to reject the entry for import if it contains one of these characters.

**Multi-value Separator**

the value to be used to separate the individual attribute values of a multi-valued attribute.

For the **Notes** connected directory, the following fields appear (see the Notes agent chapter in the *DirX Identity Connectivity Reference* for details and examples for allowed values of these fields):

**(Addr)ess Book**

the name of the Notes address book from which entries are to be exported.

**Form Name**

the IBM Notes document type to be extracted from the Notes address book.

**Item Identity Name1,2,3**

how NotesAgent matches entries in the target Notes address book with entries to be imported into the address book.

**Search Node ID**

whether (checked) or not (unchecked) NotesAgent uses the Notes identifier to match entries to be updated in the target Notes address book with entries to be imported into the address book.

**View Folder**

whether (checked) or not (unchecked) NotesAgent uses a Notes view sorted by the Notes attribute specified in the **ItemIdentityName1** field to match entries to be updated in the target Notes address book with the entries to be imported into the address book.

**Case Sensitive**

whether (checked) or not (unchecked) NotesAgent uses case-exact match when using a sorted IBM Notes view to match entries in the target Notes address book with entries to be imported.

**Replace Item**

whether (checked) or not (unchecked) existing attribute values of Notes entries in the address book are overwritten with imported values.

**Update**

whether (checked) or not (unchecked) existing Notes entries are modified with imported information or whether a new entry with the imported information is created, even if a matching entry already exists in the address book.

**Delete Entries**

whether (checked) or not (unchecked) entries that exist in the Notes address book are to be deleted if matching entries exist in the import data file.

**Multi-value Separator**

the value to be used to separate the individual attribute values of a multi-valued attribute.

**Save Org DB**

whether (checked) or not (unchecked) NotesAgent backs up the target Notes address book before performing the import operation.

**Save Server Name**

the name of a Notes server that NotesAgent is to use as a storage target when backing up a Notes address book before an import operation.

**Save DB Name**

the name of the file to which NotesAgent is to write the contents of a target Notes address book before an import operation.

**Register User**

how NotesAgent registers imported entries as IBM Notes users. Select one of the following values:

**0-DO NOT REGISTER**

do not register imported entries as IBM Notes users (default).

**1-REGISTER AND CREATE MAIL FILES**

register imported entries as IBM Notes users and create corresponding mail files immediately. Additional properties should be specified in the Register User tab sheet.

**2-REGISTER AND CREATE REQUESTS**

register imported entries as IBM Notes users and create requests for the IBM Administration Process to create corresponding mail files.

**Admin Request DB**

the name of the IBM Notes Administration Process (adminp) request database to which NotesAgent is to send request documents during *DeleteUser* changetype processing.

**Admin Request Author**

the author name of the IBM Notes Administration Process (adminp) request database to which NotesAgent is to send request documents during *DeleteUser* changetype processing.

**Path File Target Cert ID**

the pathname to the certificate ID file of a target organizational unit. The file contains the certificate that grants NotesAgent the right to create registered users for the organizational unit.

For the **NT** connected directory, the following fields appear (see the NT agent chapter in the *DirX Identity Connectivity Reference* for details and examples for allowed values of these fields):

**Data Type**

the type of imported data. Supply one of the following values:

**1-ACCOUNTS**

account data will be imported.

**2-LOCAL GROUPS**

local groups will be imported.

**3-GLOBAL GROUPS**

global groups will be imported.

**Insert RAS Info**

whether the NT account's DialinPrivilege attribute value is imported.

**Replace All Attributes**

whether NTAgent modifies only the attributes for an account delivered in the data file (unchecked), or whether it uses the defined default values for all other account attributes (checked).

**Replace All Attribute Values**

whether NTAgent can modify the attribute values of a multi-valued NT account attribute as a whole (check box marked) or not (check box unmarked).

**Replace All Group Members**

whether NTAgent adds new members to a global or local group entry (unchecked) or replaces all members in a global or local group entry (checked).

**Initial Password Only**

whether (checked) the password is only set during an add operation or whether modification is also possible (unchecked).

**Delete Entries**

whether (checked) or not (unchecked) account or group entries that exist in a WindowsNT system are to be deleted if matching entries exist in the import data file.

**Delete Group Members**

whether (checked) or not (unchecked) NTAgent deletes members from a group entry in the NT system.

**Multi-Value Separator**

the value to be used to separate the individual attribute values of a multi-valued attribute.

**Date Format**

the format of date values.

**Date Separator**

the separator character of date items.

For the **ODBC** connected directory, the following fields appear (see the ODBC agent chapter in the *DirX Identity Connectivity Reference* for details and examples for allowed values of these fields):

**Table**

the ODBC name of the table (or joined set of tables) into which entries are to be imported.

**Select By**

one or more naming attributes that the agent is to use as selection criteria during the import procedure.

**Modify Anyway**

whether the agent performs a comparison operation before modifying an ODBC entry.

**Change Type**

the alphanumeric string used in the import data file to indicate the "changetype" for ODBC entries.

**Create If Absent**

whether or not ODBCAgentImp creates a new ODBC entry ("row" in ODBC terminology) in the ODBC database if it does not find a matching entry using the naming attributes supplied in **SelectBy**.

**Insert Only**

whether or not existing entries in the ODBC database are updated with attribute values from the import data file.

**Modify**

the entry attributes in the ODBC database that the agent is to modify. Use ',' to separate multiple values.

**Relationships**

references from one table to another for which referential integrity enforcement can be handled by nullifying the reference.Use the Relationships field to permit entries ("rows" in ODBC terminology) to be deleted when entries in other tables affected by referential integrity point to them, or when it is unacceptable for the reference to a deleted entry to continue to exist.

**Always Follow References**

whether the agent follows the references defined in the Relationship field if referential integrity enforcement has not been configured in the database for the specific relationships specified.

For a detailed description of these items see the "Import Configuration File Format" sections in each DirX Identity agent chapter in the *DirX Identity Connectivity Reference*.

**Related Topics**

Tcl-based Channel

## A.13.9. Input/Output Channels

The input/output channels define the properties of the data download from the source (input) and the upload to the target directory (output).While the connected directory and the job objects hold static information, the channels contain dynamic information for the synchronization procedure.

Use this tab to assign input and output channels to a job.

**Input Channel**

the job's input channels and their related attributes.To display the properties of a selected channel, click the Properties button on the right.Use the buttons to the right of the table to add and remove new channels.

**Output Channel**

the job's output channels and their related attributes.To display the properties of a selected channel, click the Properties button on the right.Use the buttons to the right of the table to add and remove new channels.

Unless the job's agent is **metacp**, the tab also shows these properties:

**Report file**

the report file produced by the job, along with its related attributes.To display the properties of the report file, click the **Properties** button to the right.

**Ini file**

the configuration (***.ini**) file used by the job along with its related attributes.To display the properties of this configuration file, click the Properties button on the right.

**Related Topics**

Tcl-based Channel
Data File
Job

## A.13.10. Import to Identity Store Properties

The import tab depends on the script technology in use.

### A.13.10.1. Standard Scripts

Use this tab to define the parameters required to specify import and join options:

**Base Object**
the node at which the search operation that checks the attribute values of target entries starts.

**Subset**
the scope of the search operation.Supply one of the following values:

**BASE OBJECT**
check if the base object matches the search criteria, but nothing else.

**ONE LEVEL**
search for results in the given base object node and one level deeper.

**SUBTREE**
find results in the entire subtree below the given base object.

**Object Class Collection**
the set of object classes to use to create new objects. These sets are defined in the Operational Attributes tab of the connected directory (attribute Object Classes).

**Import Mode**
whether the script runs in merge or replace mode. In merge mode, the entries from the source are simply merged into the target (new entries are added, already existing ones are modified and if an operation code is contained in the source entries, these entries are deleted). In replace mode, all entries that are not contained in the source but are contained in the target will be deleted. This is done in a separate loop after the main loop. Thus replace mode makes only sense with a full result from the source!
For details about the algorithm of the script, see the section "Tcl-based Connectivity Standard Script" in the *DirX Identity Application Development Guide*. Select one of the following values:

**MERGE**
adds and modifies records in the target connected directory. Only performs deletions if an operation code is contained in the source. Can work in full and delta mode. The join filter is used for the join operation.*
REPLACE* - adds, modifies and deletes entries in the target connected directory. Requires a full result from the source (delta operation not allowed!). The Replace Filter

and the Sorting parameters are used in this case.

**Change Notification:**

If the following fields are filled, the meta controller creates notification messages about the performed changes. Set up Java-based workflows that interpret and process these messages. The generated message contains all change information and a topic with the following fields (for more information, see the topic "Understanding Notifications"):

**Object Type**

the changed object type

**Server Address**

the server's physical address

**Identity Domain**

the domain to use

**Join Expression**

the logical attribute conditions used for the identification of entries to be joined in this table. See the "Join Expressions and Filtering" section for details.
Note: only used if Import Mode is set to Merge (see also the information below).

**Replace Filter (Delete Filter)**

the filter that searches for all entries provided from this master in the target directory. The list is compared with the current list of entries to be imported into the target directory. If Delete Entries is set, no longer available entries will be deleted (thus the new list of entries replaces the previous one).

You can specify a pure LDAP search filter like \{sn=a*} here, which selects all entries with an **sn** attribute that begins with the letter **a**, or \{dxmMasterName=MyMasterName, which selects all entries for which this master is responsible.
An expression like \{dxmOprMaster=<?$src_master_name/>} uses a shortcut reference previously defined in the control script of the workflow. This generic expression retrieves the dxmMasterName property from the source directory and searches for all entries with dxmOprMaster set to this value.
Note: As described, this field can contain references. See the chapter "Customizing Object References" in the *DirX Identity Customization Guide* for more information.



Only used if Import Mode is set to Replace.

**Sorted list**

whether (checked) or not (unchecked) the resulting entry list should be sorted. This function is only used if Import Mode is set to Replace (to sort the list retrieved by the Replace Filter).

**Sort/Join key**

the attribute used for sorting and comparison criteria (join criteria). Specify a unique field.

**Sort/Join order**

whether the results are sorted in **ASCENDING** or **DESCENDING** order.

**Read DN only**

whether (True) or not (False) to enable optimization for lower memory consumption. Note: This switch was previously named **Perform Read**.

**FALSE**

searches complete entries with all attributes during the first search. No additional search operations are necessary later on. This requires a lot of memory but results in maximum speed.

**TRUE**

searches only DNs during the first search and later on all other attributes for each entry by an additional search operation. This lowers memory consumption enormously but decreases performance by more than 50 %.

**Paged Read**

whether (checked) or not (unchecked) the script works in paged mode, if the mode is supported by the LDAP server.

**Page Size**

the size of the pages. Default is 1000 bytes.

##### A.13.10.1.1. Join Expressions and Filtering

The **Join Expression** field is only used when **Import Mode** is set to **Merge**. This control allows you to set simple combinations of filters with a table-like structure (simple mode). Alternatively, you can set a collection of expert filters (expert mode). Use the right-most upper button to switch between the two modes.

In **simple mode**, the control allows you to define join conditions in the following way (a cell grid is shown):

- You can enter attributes into each cell of the table.
- Each line defines a separate join expression where the attributes are combined by an and condition (for example: Surname & GivenName & TelephoneNumber).
- The expressions (lines) are evaluated from top to bottom. Each expression is used as a filter condition for a search in the directory.
- When no hit or more than one hit occurs, then the next expression (line) is used.
- Exactly one hit stops the evaluation. The found entry is used to join the information to be imported into it.
- When all lines have been evaluated and no single entry can be identified, the DN is used. If this search also fails, a new entry must be created in the directory (only if **Exact Action** = FALSE).

In **expert mode**, the control allows you to define join conditions in the following way (several rows with only one cell are shown):

- You can enter an expert filter condition into each row of the table. Set the filter expression like
  {(employeeNumber=[lindex $rh_ldap(dxmNTuserName) 0])}
  which means that the first item of the mapped rh_ldap(dxmNTuserName) field is compared with the employeeNumber field in the LDAP directory. Of course you can define more complex join criteria.
- Each line defines a separate join expression.
- The expressions (lines) are evaluated from top to bottom. Each expression is used as a filter condition for a search in the directory.
- When no hit or more than one hit occurs, the next expression (line) is used.
- Exactly one hit stops the evaluation. The found entry is used to join the information to be imported into it.
- When all lines have been evaluated and no single entry can be identified, the DN is used. If this search also fails, a new entry must be created in the directory (only if **Exact Action** = FALSE).
- Note: These fields can contain references. See the chapter "Customizing Object References" in the *DirX Identity Customization Guide* for more information.

**Related Topics**

Tcl-based Channel
Entry Handling

#### A.13.10.2. Previously Used Scripts

Previous scripts used two different tabs for join and import properties.

Join and import properties are needed to set up the specific options for importing data into the Identity store. The join parameters describe how to identify a target entry by the given source entry data. Other import parameters specify search operations for the relevant target entries to be updated. Another subset of parameters describes the restrictions during data import.

Use this tab to define all parameters needed to specify import options.

The subsequent items define import parameters:

**Use DN**

whether (checked) or not (unchecked) the distinguished name is used for joining a source and a target entry.

**Superior info**

information that permits the necessary superior nodes for an entry to be constructed if they do not exist when the entry is added to the Identity Store.

**Base Object**

the node at which the search operation that checks the attribute values of target entries starts.

**Subset**

the scope of the search operation. Supply one of the following values:

**BASE OBJECT**

check if the base object matches the search criteria but nothing else.

**ONE LEVEL**

search for results in the given base object node and one level deeper.

**SUBTREE**

Find results in the entire subtree below the given base object.

**Filter**

the matching criteria. This can be any logical expression in LDAP notation. Use "&" for "AND", "|" for "OR", and "!" for "NOT", as well as "=" for equality check, "~=" for phonetic equality check, and "⇐" for less, ">=" for greater.

**Filter check**

the expression used in the Tcl program to check the validity of the given filter.

**Sorted list**

whether (checked) or not (unchecked) the resulting entry list should be sorted.

**Sort key**

the attribute that is used for sorting.

**Sort order**

whether the results are sorted in **ASCENDING** or **DESCENDING** order.

**Remove objects**

whether (checked) or not (unchecked) entries which are no longer valid can be deleted immediately.

**Marking attribute**

the attribute name used to indicate that an entry can be deleted. It allows another connected directory to check the data before the entry is really deleted.

**Marking value**

the value for the **Marking attribute**.

**Keep objects**

the list of distinguished names of entries which must not be deleted in any case.

**Allow rename**

whether (checked) or not (unchecked) the distinguished name of the object can be modified.

**Create new entries**

whether (checked) or not (unchecked) creation of new entries is allowed.

**Perform read**

whether (checked) or not (unchecked) the system will read all attributes during a search operation.It may however be sufficient just to read the distinguished name for the first, and later query for further attributes, if necessary.

**Process all attributes**

whether (checked) or not (unchecked) all attributes of the attribute configuration should be processed, not just the selected ones (not yet implemented).

**Related Topics**

Tcl-based Channel
Join to Identity Store Properties

## A.13.11. Join to Identity Store Properties

Join and import properties are needed to set up the specific options for importing data into the Identity store.The join parameters describe how to identify a target entry by the given source entry data.Other import parameters specify search operations for the relevant target entries to be updated.Another subset of parameters describes the restrictions during data import.

Use this tab to define all parameters needed for performing a join operation.

The following items are necessary for join operations:

**Join Expression**

the logical attribute conditions used for the identification of entries to be joined in this table.See the following sections for details.

**Join object class**

the connected-directory specific object class used to identify an attribute set of a particular connected directory.During the join operation, this object class can be assigned to the target entry, which enables the synchronization of the attributes associated with this class.

**Matching attribute**

the attribute to be used as an identifier for the master directory of the current (sub-)set of attribute values assigned to the target entry. The provider of the value (sub-)set can store its name into this attribute. Other connected directories providing the same data types can check to see if their name is stored in this attribute, and then write their attribute values to the Identity store only in this case.

**Matching value**

the value of **Matching attribute**.

The join table **Join Expression** in the Join Properties tab allows you to define join conditions in the following way:

- You can enter attributes into each cell of the table.

- Each line defines a separate join expression where the attributes are combined by an and condition (for example: Surname & GivenName & TelephoneNumber).
- The expressions (lines) are evaluated from top to bottom. Each expression is used as a filter condition for a search in the directory.
- When no hit or more than one hit occurs, the next expression (line) is used.
- Exactly one hit stops the evaluation. The found entry is used to join the information to be imported into it.
- When all lines have been evaluated, the DN is used. If this search fails also, a new entry must be created in the directory.



When the schema has not been read from the LDAP directory, two questions marks are appended to the value of the object type field. You simply must synchronize the schema to solve this problem.

**Related Topics**

Tcl-based Channel
Import to Identity Store Properties

## A.13.12. Other Properties of a Channel

Use this tab to display additional properties of a channel configuration object. These properties are mainly file-related and correspond to the file assigned to the connected directory to which the channel is connected.

**Selected File**

the file item object that describes the data file to be accessed for bulk data read or write operations. Use the Properties button to view the properties of the file item or the Delete button to delete the value or the Browse button to change the value.

**File Mode**

the mode in which the file is opened. Select one of the following values:

**READ**

the file will be opened for read access (this means that the channel is an input channel).

**WRITE**

the file will be opened for write access (this means that the channel is an output channel).

**APPEND**

same as WRITE, but the previous content of the file is kept and the new data are just appended to the end of the file.

**Sorted List**

whether (checked) or not (unchecked) the result of this channel is to be sorted.

For output channels: the system produces a bulk data file with a sorted list of entry items.
For input channels: the entries are read into an internal sorted list. Use the **Sort/Join Key** and **Sort/Join Order** fields to specify the details.

**Sort/Join Key**

the attribute used for sorting.

**Sort/Join Order**

the sequence to sort (ascending or descending order).

**To Upper**

whether (checked) or not (unchecked) to convert all input and output field values to uppercase.

**Target Entry Exists**

the attribute name that is used to determine whether an entry in the target system already exists. This field is only necessary for two-step workflows, where the meta controller cannot check directly whether an entry exists in the target system. See the "Calculate Action" section in the *DirX Identity Application Development Guide* for details.
Example:

> Your target system is a mail system. Your workflows create entries (mail boxes) in the target system and then transfer the generated mail address back to the directory and populate the mail attribute there. In this case, a populated mail attribute (an existing mail attribute) indicates that the entry already exists at the target system side. The scripts can use this value to determine whether an entry must be created or only modified.

**Related Topics**

Tcl-based Channel
File Item

## A.13.13. Register User

The Register User properties are additional properties for the import of user entries in an IBM Notes administration database.

Use this tab to specify these properties. The following items appear:

**Item Mailbox Name**

the attribute to use as the mailbox name.

**Item User ID**

the attribute to use as the User ID.

**Path File Cert ID**

the path to the certificate ID file **cert.id**, which is a binary file that is supplied with the IBM Notes Server installation software. This file contains the certificate that grants NotesAgent the right to create registered users.

**Path User ID**

the directory in which NotesAgent is to store IBM Notes user IDs created during the user registration process.

**Registration Server**

the name of the Notes registration server that is to register the users in the Notes server address book.

**Mail Server**

the name of a Notes server on which NotesAgent is to create user mailboxes during the user registration process.

**Min(imal) Password Length**

the minimum number of characters that a user password must have.

**Create Address Book Entry**

whether or not NotesAgent creates Notes entries in the target Notes address book for IBM Notes users that it registers during the import process.

**Create Mail Database**

whether or not NotesAgent creates user mailboxes for IBM Notes users that it registers during the import process.

**Create ID File**

whether or not NotesAgent creates a user ID file for IBM Notes users that it registers during the import process.

**Save ID in Address Book**

whether or not NotesAgent saves the user ID files it creates as attachments of the Notes entries for the registered users.

**Save ID in File**

whether or not NotesAgent saves the user ID files it creates in individual files.

**Create North American ID**

whether or not NotesAgent creates United States security encrypted User ID files.

**Client Type**

the type of IBM Notes client that NotesAgent is to associate with the registered users it creates during the import process.

For a detailed description of the items, see the *DirX Identity Connectivity Reference*.

**Related Topic**

## A.13.14. Selected Attributes

This tab displays the Selected Attributes Editor, which you use to define the set of attributes to be synchronized between the source and target connected directories.The Selected Attributes Editor is accessible through the workflow configuration wizard in the Global View and through the channel configuration object in the Expert View.

The Selected Attributes Editor consists of two tables:

**Attribute configuration**
: the attributes in the attribute configuration that represents the schema

**Selected attributes**
: the attributes to be synchronized and their respective synchronization flags

Use the arrow buttons to move attributes between the attribute configuration list and the selected attributes list.To move an attribute, select it in the list and then click the appropriate button.To add synchronization flags to a selected attribute, click in the row the Flags column and then click the ellipsis (...) button to open the Synchronization Flags dialog.Check or uncheck a synchronization flag to enable or disable it.For a description of the possible synchronization flags, see the section "Using the Selected Attribute Editor" in the chapter "Using the DirX Identity Manager" in the *DirX Identity User Interfaces Guide*.



Only selected attributes can be used for a mapping operation. If you remove an attribute in the Selected Attribute Editor, it is marked red in the Mapping Editor to indicate that it is no longer available.



*Notes for specific workflows:*

**All Two-Step Standard Workflows**

In two-step standard workflows, four channels exist. Only the two **metacp** channels are used for configuration. To indicate this, the other two channels for the agent do not contain selected attributes (the right pane of the form is empty).

**Meta2DSML**

DDN and objectClass must be the first and second entry in the selected attributes list.

## A.13.15. Superior Info

DirX Identity sometimes needs to create new entries where the higher-level nodes in the target directory do not exist.The Superior Info Editor allow you to specify all of the required information for creating these objects in a Superior Info definition.DirX Identity will then create the higher-level nodes before it adds the entry.

The Superior Info Editor provides three table columns: **Naming attribute**, **Mandatory**

**attribute** and **Default value**.To add a table row, click the button on the right and then enter the information.To delete a row, click .For more information on a Superior Info definition and how to use the editor, see the section "Using the Superior Info Editor" in the chapter "Using DirX Identity" in the *DirX Identity User Interfaces Guide*.

## A.13.16. Tcl-based Workflow

A Tcl-based workflow consists of one or more activities that carry out a data synchronization operation between two connected directories.A workflow configuration object describes the configuration information for a particular workflow.Use the workflow configuration object to define synchronization workflows.

DirX Identity currently supports any number of sequential or parallel activities.Activities can represent jobs or workflows.Linking an activity to a workflow allows you to build nested workflow structures.Activity structures within one workflow can be either pure sequential or in the form of a tree structure (an activity can have 1 to *n* successors but only one predecessor).A combination of nested workflows and parallel activities allows you to build almost any required workflow structure:

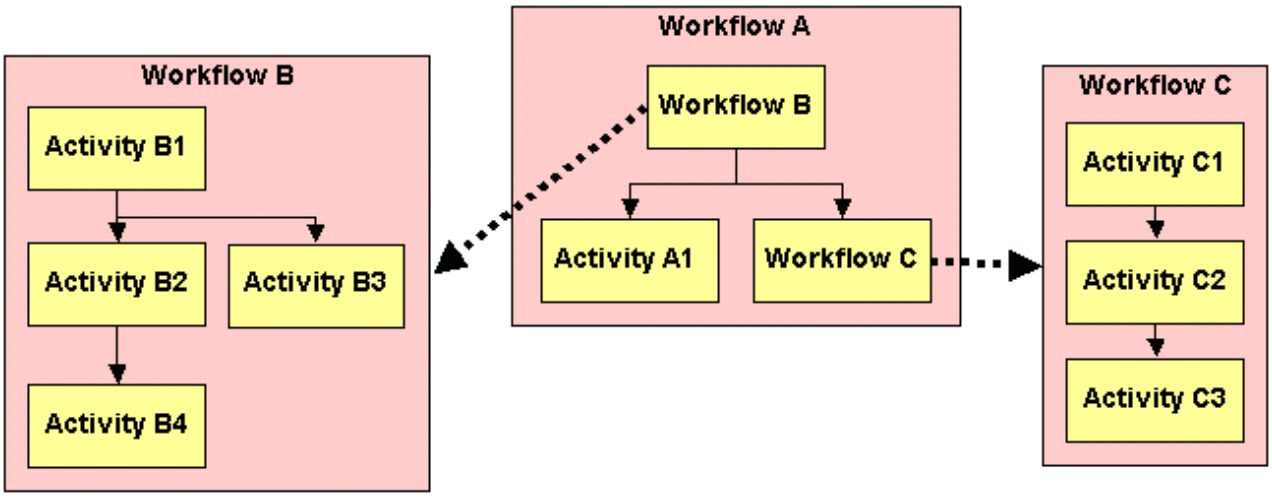



*Figure 32. Nested Workflow with Parallel Activities*

When workflow A starts, it starts as first activity workflow B. After running activity B1, activities B2 and B3 are started in parallel.When activity B2 ends, activity B4 is started.Workflow B ends when activities B3 and B4 are completed.Now activity A1 and workflow C are started.Workflow C runs three activities (C1 to C3) in sequence.When activity A1 and workflow C are both completed, workflow A is finished.

when nested workflows or parallel activities are not configured correctly, no status entry is written. The errors are written to the log files or the event log (on Windows).

you can use a specific job instance only once in a workflow definition. It is not allowed to use the same job from two different activities.

In DirX Identity, data is always transferred through connected directories. For example, if

you want to establish a workflow that contains two activities-one for export from the source connected directory and one for import to the target connected directory-and these activities exchange data through a file, you must set up a connected directory that represents the file.

As you can see from the previous example, connected directories can reside at the end points of a workflow or between the workflow's activities. Connected directories that reside between activities are "intermediate" connected directories and are usually file-based. An intermediate connected directory contains the name of a file generated by a particular synchronization activity.

A workflow has a control flow (the sequence of activities) and a data flow (the connection of data inputs with data outputs) and these two flows can be different. For example, if a workflow has three activities that run in sequence, the last one could use data from the first one, which is not visible from the control flow.

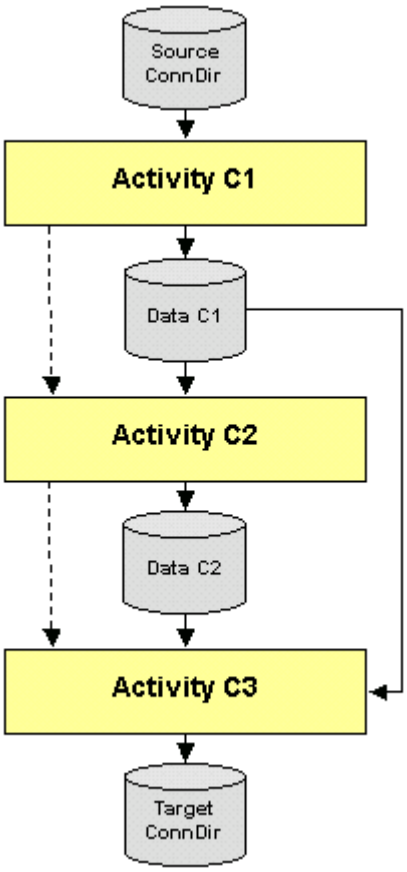



*Figure 33. Control and Data Flow*

Activity C3 uses data from activity C2 and activity C1, which is not visible from the control flow (dotted lines).

You can start configured workflows in different ways. See the section "Starting Tcl-based Workflows" for details. The workflow engine (WFE) component of the C++-based Server runs workflows and the activities it contains.

Use this tab to assign properties to a workflow. The items shown in this tab are:

**Name**

the name of the workflow.

**Description**

the description of the workflow.

**Version**

the version number of the workflow.

**C++-based Server**

the C++-based Server associated with the workflow. To display its properties, click the Properties button on the right.

**Status Life Time**

the maximum time that status entries and related files will be retained after a run of the workflow. This field allows you to redefine the default value set in the Configuration object to a specific value for this workflow. An empty field means that the default value of the Configuration object is used.

**Status Compression Mode**

the detail level and number of status messages generated. This switch can help to reduce the load on the status tracker or simply to avoid uninteresting status entries. The following levels are available:

**(None)**

the **default behavior** of this feature is inherited from the central configuration object.

**0 - Detailed**

detailed messages are sent during the workflow lifetime (compatibility mode, default)

**1 - Compressed**

status messages are collected during the workflow run as far as possible and sent at the very end of a workflow. This reduces the number of status messages by 50 % or more.

**2 - Minimized if OK**

only a workflow status entry is generated at the very end of a workflow if the workflow ended with status OK. No activity status entries are generated and no data is copied to the status area.

**3 - Suppressed if OK**

no status information is created at all and no data is copied to the status area if the workflow ended with status OK.

**Server Event Support**

whether (checked) or not (unchecked) the workflow contains an activity that can react to server events; for example, to "keep alive" requests or shutdown requests. As an example, the supervisor uses these messages to control the event manager workflow.

**Wizard**

the link to the wizard associated with the workflow, or an empty field if no wizard exists for the workflow.



We recommend that the name of the wizard shall contain the types of directories at the endpoints of the workflow (for example, LDAP and FILE). You can include other information between these endpoint values (for example, LDAP-3step-FILE) to identify a more specific wizard.

**Source Directory**

the link to the source connected directory to indicate the start of the workflow. This field allows for evaluating the direction arrow if both connected directories at the endpoints are of the same type (for example, the value of **Endpoints** is LDAP-LDAP).

**Endpoints**

the directory type endpoint (for example, LDAP and FILE). This information allows DirX Identity to determine which workflows fit between the source and target connected directories that are connected with a workflow line. Calculation is performed automatically whenever the workflow object is changed, but you can enforce it at any time by clicking the Refresh button on the right if it is not correct (for example, if you changed the channel of a job to point to another type of connected directory).

**Bidirectional**

whether (checked) or not (unchecked) both arrows on a workflow line in the Global View are set when this workflow is part of a workflow line.

**Related Topics**

Tcl-based Activities
(Central) Configuration
C++-based Server
Service
Specific Attributes
Schedule
Workflow - Operation

## A.13.17. Workflow - Operation

Use this tab to set the checkpointing parameters.The items shown in this tab are:

**Enabled**

the workflow works in checkpoint mode.One or more jobs must also be enabled for checkpointing.

**Retry Limit (default is 3)**

the maximum number of retries.If the limit is reached, the workflow runs again in complete full or delta mode (as defined).

Workflow parameters that display the checkpoint status are:

**Retry Count**

the number of retries already performed.

**Restart Activity**

one or more activities that are started in parallel during a retry operation. This field is read-only.

**Related Topics**

Tcl-based Activities
(Central) Configuration
C++-based Server
Service
Specific Attributes
Schedule
Workflow - General Properties

## A.13.18. Workflow - General Properties

Use this selection in the workflow wizard to supply the workflow data that is common to all workflow objects.

**Name**

the name of the workflow.

**Description**

a description for the workflow.

**Version**

the version number of the workflow.

If you have copied the workflow from a default application, you can enter new values into these fields.

**Related Topics**

Tcl-based Workflow
Workflow Line

## A.13.19. Workflows

A folder for the workflow configuration objects in the configuration database.

**Name**

the name of the folder.

**Description**

descriptive text for this object.

Within a property page, the content of this folder is shown as a pop-up list of a combo box:

**Workflow**

the workflow object currently used by the object whose properties are displayed.Use the arrow button to display the list of all available workflow objects in the workflow folder.Use the Properties button to display the properties of the currently selected workflow object.

**Related Topic**

Workflows

# Appendix B: Mapping Functions

This section provides reference information for the mapping functions, presented in alphabetical order.

## B.1. addAttributes

Returns a variable that contains an ADD attribute definition for an LDIF change file.

Synopsis:

**addAttributes** *value1 value2 ...*

Description:

Returns a list containing the string ADD-VALUE indicating to add the following values.

## B.2. deleteAttributes

Returns a variable that contains a DELETE attribute definition for an LDIF change file.

Synopsis:

**deleteAttributes** *value1 value2 ...*

Description:

Returns a list containing the string DELETE-VALUE indicating to delete the following values.

## B.3. hdmsCmd2dmsid

Extracts the DMS Identifier from the CMD field of an HDMS record.

Synopsis:

**hdmsCmd2dmsid** *cmdfield*

Description:

This function extracts the DMS identifier from the first field of an HDMS record (*cmdfield*).

Example:

> The function applied on cmdfield="INSERT;PERS;29874" evaluates to
>
> 29874

## B.4. hdmsdata2dn

Creates a DN from an HDMS entry.

Synopsis:

**hdmsdata2dn** *handleName*

Description:

By default, this function uses the field's name (abbreviation NAME), Christian name (abbreviation CHN), org1 (abbreviation ORG1), company (abbreviation O) and country (abbreviation C) from an array that is defined by *handleName* to create a distinguished name.

Example:

With NAME=Mayer, CHN=John, ORG1=development, O=My-Company and C=de the resulting distinguished name is:

cn=Mayer John,ou=development,o=My-Company,c=de

## B.5. hdmsData2telno

Gets a telephone or a facsimile telephone number as a concatenation of HDMS attributes.

Synopsis:

**hdmsData2telno** *country extarea extnet extext*

Description:

This function concatenates its four input arguments to a canonical four-part telephone (or fax) number with the blank delimiter. Using an internal table, the country is converted into the country code.

Example:

For instance, the input parameters country="DE", extarea="89", extnet="777" extext="12345" evaluates to

+49 89 777 12345

## B.6. ifEqual

Checks a variable's value and sets the result to one of two values.

Synopsis:

**ifEqual** *variable value then-value else-value*

Description:

If *variable* equals *value*, the result is set to *then-value*, else to *else-value*.



This function works on a single-valued list only.

## B.7. ifNotEqual

Checks a variable's value and sets the result to one of two values.

Synopsis:

**ifNotEqual** *variable value then-value else-value*

Description:

If *variable* does not equal *value*, the result is set to *then-value*, else to *else-value*.



This function works on a single-valued list only.

## B.8. lADSpathCreate

Creates an ADS path.

Synopsis:

**lADSpathCreate** *prefix type1 value1 type2 value2 ...*

Description:

This routine is used to generate an ADSpath from a list of input field pairs. This routine is useful for mappings for import to ADS / Exchange. In order to be very flexible, the user needs to pass the types of the components of the ADS path, too.

The first argument *prefix* allows defining a constant value as the first part of the path.

Each of the next pair of arguments is combined to a part of the path (for example "cn" and ldap.cn with a value of 'Huber Max' are combined to 'cn=Huber Max'. If the value field is empty, this part of the DN is omitted.

Examples:

The arguments

"LDAP://server1:389" "o" ldap.o "ou" ldap.ou "cn" ldap.cn

create with values
ldap.cn=Huber Max
ldap.ou=Development
ldap.o=My-Company

an ADS path: LDAP://server1:389/o=My-Company/ou=Development/cn=Huber Max

Error Handling

An error is returned in the global variable **mapping_error** when

- One of the input values is a multi-valued string

# B.9. lBaseDNreplace

Replaces the base DN in a DN.

Synopsis:

**lBaseDNreplace** *string string1 string2*

Description:

Replaces *string1* by *string2* in the string array *string*. Replacement is done starting at the end of the string.

# B.10. lBool2Integer

Converts boolean values (TRUE and FALSE) to integer values (1 and 0).

Synopsis:

**lBool2Integer** *boolean*

Description:

FALSE will be mapped to "0". TRUE will be mapped to "1". For all other values, the value "" is generated.

# B.11. lDate2GMT

Converts an list of dates into a list of GENERALIZED time strings.

Synopsis:

**lDate2GMT** *date_list*

Description:

This routine converts a list of dates into a list of generalized time strings (the LDAP format

for date and time information).

These input formats can be converted:

JJJJMMDDhhmmssZ (generalized time format is a valid input format)
MM/DD/JJJJ hh:mm:ss
DD.MM.JJJJ hh:mm:ss
JJJJ-MM-DD hh:mm:ss

Examples:

```
20030228123822Z → 20030228123822Z
02/28/2003 12:38:22 → 20030228123822Z
28.02.2003 12:38:22 → 20030228123822Z
2003-02-28 12:38:22 → 20030228123822Z
```

Error Handling

An error is returned in the global variable **mapping_error** when

- An invalid input date format is detected.
- The date is outside the range from 1902-01-01 00:00:00 to 2037-12-31 00:00:00 (Tcl restriction).

## B.12. lDNcreate

Creates a DN.

Synopsis:

**lDNcreate** *type1 value1 type2 value2 ... [typen valuen] | valuen*

Description:

This routine is used to generate a distinguished name (DN) from a list of input field pairs. This routine is useful for import mappings. In order to be very flexible, the user needs to pass the types of the components of the DN, too.

Each of the pairs of arguments is combined to a part of the path (for example "cn" and ldap.cn with a value of 'Huber Max' are combined to 'cn=Huber Max'). The pair results are concatenated with commas.

When the number of arguments is a multiple of two, the path ends with a combination of type=value. When the number of arguments is not a multiple of two, the path ends with a constant (for example 'o=pqr'). If the value field is empty, this part of the DN is omitted.

Examples:

```
The arguments
```

"cn" ldap.cn "ou" ldap.ou "o" ldap.o

create with values
ldap.cn=Huber Max
ldap.ou=Development
ldap.o=My-Company

as result the DN:

cn=Huber Max,ou=Development,o=My-Company

"cn" ldap.cn "ou" ldap.ou "l=DE,o=pqr"

create with the same values as result the DN:

cn=Huber Max,ou=Development,l=DE,o=pqr

Error Handling

An error is returned in the global variable **mapping_error** when

- one of the input values is a multi-valued string

## B.13. lDNsplit

Splits a DN into the elements of a Tcl array.

Synopsis:

**lDNsplit** *dn arrayname*

Description:

This routine is used to split a distinguished name (DN) into the elements of a Tcl array. The first argument *dn* is split into elements by using the comma separator. These elements are split again into the element name and value.

Note that the result of this function is a return code and not the array. The return code is 0 if no errors occurred and 1 if syntax errors were detected.

Examples:

If DDN contains the value "cn=Huber\,Hans,ou=ICN,o=My-Company"

lDNsplit DDN "my_array"

creates a Tcl array with these elements:

cn=Huber\,Hans
ou=ICN

o=My-Company

You can access the elements with my_array(cn), my_array(ou), ...

Error Handling

An error is returned in the global variable **mapping_error** and in the return code when

- An invalid syntax is detected (for example "abc/def") and results in a return code of '1'.

When an element in the DN occurs multiple times (for example, "cn=Huber,ou=SNS,ou=ICN,o=My-Company"), a number is added automatically to the field name:

cn=Huber
ou=SNS
ou1=ICN
o=My-Company

When accessing the array elements, you must be sure that they exist. Otherwise a Tcl error will be the result. You can test the availability of such an element with **info exists**.

## B.14. lInteger2Bool

Converts integer values (0 and 1) to boolean values (FALSE and TRUE).

Synopsis:

**lInteger2Bool** *integer*

Description:

"0" will be mapped to FALSE. "1" will be mapped to TRUE. For all other values the value "" is generated.

## B.15. listFirst

Returns the first element of a list as a string.

Synopsis:

**listFirst** *list*

Description:

Returns the first element of a list as a string.

## B.16. listLast

Returns the last element of a list as a string.

Synopsis:

**listLast** *list*

Description:

Returns the last element of a list as a string.

## B.17. lListAppend

Appends elements to a list.

Synopsis:

**lListAppend** *list list ...*

Description:

Appends all elements to a list.

## B.18. lListFirst

Returns the first element of a list as a new single-valued list.

Synopsis:

**lListFirst** *list*

Description:

Returns the first element of a list as a new single-valued list.

## B.19. lListLast

Returns the last element of a list as a new single-valued list.

Synopsis:

**lListLast** *list*

Description:

Returns the last element of a list as a new single-valued list.

## B.20. lListNth

Returns the *n*th element of a list as a new single-valued list.

Synopsis:

**lListNth** *list n*

Description:

Returns the *n*th element of a list as a new single-valued list.

## B.21. lListRest

Returns the second to last element of a list.

Synopsis:

**lListRest** *list*

Description:

Returns the second to last element of a list as a new list.

## B.22. lPA2String

Replaces the $ characters with carriage returns.

Synopsis:

**lPA2String** *list*

Description:

Replaces the $ characters in a postal address field with carriage returns. Replacement is done for all the elements of the Tcl list.

## B.23. lString2PA

Replaces carriage returns with $ characters.

Synopsis:

**lString2PA** *list*

Description:

Replaces carriage returns with $ characters in a postal address field. Replacement is done for all the elements of the Tcl list.

## B.24. lStringAppend

Appends strings to all elements of a list.

Synopsis:

**lStringAppend** *list string1 string2 ...*

Description:

This routine takes the given list and appends to all its elements the given strings *string1, string2, ....*

The result value of this routine is a Tcl list with all the appended values.

Note: The native TCL command append doesn't work properly for Tcl lists. Therefore the wrapper function is needed.

## B.25. lStringCompose

Composes an output string from a variable number of input strings.

Synopsis:

**lStringCompose** *separator string1 string2 ...*

Description:

The output string is composed of all values of the input strings (*string1 ... string n*) separated by the defined *separator*. Be sure to have strings and not Tcl lists as input parameters (you can use, for example, the ListFirst function to extract the first string of a Tcl list).

If an input string has no value, it is simply omitted together with the separator value.

Examples:

```
lStringCompose ";" "abc" "def" "ghi" results in "abc;def;ghi"

If sn="Huber", givenName="" and employeeNumber="1234" then

lStringCompose " " sn givenName employeeNumber

results in "Huber 1234"
```

## B.26. lStringConvertChars

Replaces UTF-8 characters with underlying vowels.

Synopsis:

**lStringConvertChars** *value*

Description:

lStringConvertChars replaces certain UTF-8 characters with their underlying vowels or other characters. This routine is useful for attributes that have a limited character set only; for example, PrintableString (as used in the attribute "mail").

The function takes a Tcl list of strings as input, performs the necessary replacements and returns a new Tcl list of the modified strings.

The following characters are substituted by the underlying vowel:

À Á Â Ã Å Æ -> A

à á â ã å æ -> a

Ä -> AE

ä -> ae

È É Ê Ë -> E

è é ê ë -> e

Ì Í Î Ï -> I

ì í î ï -> i

Ò Ó Ô Õ Ø -> O

ò ó ô õ ø -> o

Ö -> OE

ö -> oe

Ù Ú Û -> U

ù ú û -> u

Ü -> UE

ü -> ue

The following characters are substituted by an equivalent character:

Ç -> C

ç -> c

ß -> ss

Ñ -> N

ñ -→ n

Ý -→ Y

ý ÿ -→ y

Examples:

Müller -→ Mueller

Résidence -→ Residence

# B.27. lStringEncrypt

Encrypts a Tcl list and returns the encrypted values as Tcl list.

Synopsis:

**lStringEncrypt** *list*

Description:

This routine takes the given list *list* and encrypts each of its values.

Use this function, for example, to generate default passwords that must be transferred to the target system in encrypted format. You can also use the function to encrypt data values before you write them into the directory.

See the user hook function uh::prolog description in the section "Understanding Tcl-based Workflows" in the *DirX Identity Application Development Guide* for more information how to initialize the certificate information.

Error Handling:

An error is returned in the global variable **mapping_error** and in the return code when

- Encryption of one of the values fails (results in a return code of '1').

# B.28. lStringEscape

Escapes the characters **;\{}** in all elements of the list.

Synopsis:

**lStringEscape** *list*

Description:

Escapes the characters **;\{}** in all elements of the list. This function is useful before working in Tcl with these variables. After other operations, the escaped character sequences can be

removed with the reverse function lStringUnescape.

Note: This escaping is necessary because **metacp** uses some characters in a special way. The character ";" serves as a multi-value separator. The characters "\{" and "}" are needed to handle Tcl lists correctly and prevent unequal numbers of these characters in attribute values.

# B.29. lStringEscapeLDIF

Converts elements of an LDIF content file or LDIF change file.

Synopsis:

**lStringEscapeLDIF** *value ldif_opcode*

Description:

Escapes the characters ;\{} in all elements of the list. This function is useful before working in Tcl with these variables. After other operations, the escaped character sequences can be removed with the reverse function lStringUnescape.

Note: This escaping is necessary because **metacp** uses some characters in a special way. The character ";" serves as a multi-value separator. The characters "\{" and "}" are needed to handle Tcl lists correctly and prevent unequal numbers of these characters in attribute values. In contrast to the mapping function lStringEscape, which is only appropriate for Tcl lists representing the full information of an attribute value, the function lStringEscapeLDIF is also applicable to lists representing LDIF changes.

If ldif_opcode is not "MODIFY", then behavior with respect to values is the same as for the lStringEscape mapping function.

Examples:

The following example session illustrates the difference between lStringEscapeLDIF and lStringEscape.

metacp> set x [llist ADD-VALUE "A\;B c"]

results in:

ADD-VALUE {A;B c}

metacp> set bad_mapping [lStringEscape $x]

results in

{ADD-VALUE \{A\;B c\}}

metacp> set good_mapping [lStringEscapeLDIF $x MODIFY]

results in

{ADD-VALUE {A\;B c}}

We can see the following information from this example:

1. The source list x represents an attribute change **Add value "A;B c"**
2. The list bad_mapping represents an attribute change **Add values "{A;B", "c}** (with ; and {} being escaped), that is inappropriate here.
3. The list bad_mapping represents an attribute change **Add value "A;B c"** (with ; being escaped) that is correct.

# B.30. lStringEscapeVar

Escapes the defined characters in all elements of the list.

Synopsis:

**lStringEscapeVar** *list chars*

Description:

Escapes the defined characters *chars* in all elements of the list. This function is useful before working in Tcl with these variables. After other operations, the escaped character sequences can be removed with the reverse function lStringUnescapeVar.

# B.31. lStringModify

Replaces either '*n*' or all occurrences of string A with string B in all elements of a list.

Synopsis:

**lStringModify** *list string1 string2 ?n?*

Description:

This routine takes the given *list* and replaces (*n* times) in each of its strings the substring *string1* with substring *string2*.

If *n* is missing or set to 'all', then all of the matching substrings will be replaced.

The result value of this routine is a Tcl list with all the replacements made.

Examples:

lStringModify {{The weather is not fine} {It's raining}} "i" "X" 1

will return the following Tcl list: {{The weather Xs not fine} {It's raXning}}

lStringModify {{The weather is not fine} {It's raining}} "i" "X" 2

will return the following Tcl list: {{The weather Xs not fXne} {It's raXnXng}}

## B.32. lStringPrefix

Adds a string before each element of a Tcl list.

Synopsis:

**lStringPrefix** string *list*

Description:

This routine takes the given *list* and prefixes each of its elements with the string *string*.

The result value of this routine is a Tcl list with all elements prefixed.

Examples:

lStringPrefix "cn=" {{Huber} {Maier}}

will return the following Tcl list: {{cn=Huber} {cn=Maier}}

## B.33. lStringRange

Returns all the characters of each element in the list in the range from first to last.

Synopsis:

**lStringRange** *list first last*

Description:

The routine operates in the same way as the original Tcl function string range, except that it works for Tcl lists.

It returns all the characters of *list* in the range from *first* to *last* (inclusive).

## B.34. lStringTrim

Drops leading and trailing characters from all elements of a list.

Synopsis:

**lStringTrim** *list ?chars?*

Description:

The routine operates in the same way as the original Tcl function string trim, except that it

works for Tcl lists.

It drops leading and trailing characters from a string. If *chars* is not specified, then white spaces are removed (spaces, tabs, newlines, and carriage returns).

## B.35. lStringTrimLeft

Drops leading characters from all elements of a list.

Synopsis:

**lStringTrimLeft** *list ?chars?*

Description:

The routine operates in the same way as the original Tcl function string trimleft, except that it works for Tcl-Lists.

It drops leading characters from a string. If *chars* is not specified, then white spaces are removed (spaces, tabs, newlines, and carriage returns).

## B.36. lStringTrimRight

Drops leading and trailing characters from all elements of a list.

Synopsis:

**lStringTrimRight** *list ?chars?*

Description:

The routine operates in the same way as the original Tcl function string trimright, except that it works for Tcl lists.

It drops trailing characters from a string. If *chars* is not specified, then white spaces are removed (spaces, tabs, newlines, and carriage returns).

## B.37. lStringUnescape

Unescapes the characters **;\{}** in all elements of the list.

Synopsis:

**lStringUnescape** *list*

Description:

Unescapes the characters **;\{}** in all elements of the list. This function is useful after working in Tcl with these variables. The reverse function is lStringEscape.



This escaping is necessary because **metacp** uses some characters in a special way. The character ";" serves as a multi—value separator. The characters "\{" and "}" are needed to handle Tcl lists correctly and prevent unequal numbers of these characters in attribute values.

## B.38. lStringUnescapeVar

Unescapes the defined characters in all elements of the list.

Synopsis:

**lStringUnescapeVar** *list chars*

Description:

Unescapes the defined characters *chars* in all elements of the list. This function is useful after working in Tcl with these variables. The reverse function lStringEscapeVar.

## B.39. lWordCapitalize

The first character of all words is converted to uppercase, the rest in lowercase for all elements in the list.

Synopsis:

**lWordCapitalize** *list*

Description:

Converts the first character of each word in all elements in the list to uppercase and all other characters to lowercase.

## B.40. lWordFirst

Retrieves the first word for all elements of the list.

Synopsis:

**lWordFirst** *list*

Description:

This routine takes the given list (list of strings) and extracts the first word from each of the strings. The result value of this routine is a list that contains the first word of each input string.

Example:

```
lWordFirst {{The weather is not fine} {It's raining}}
```

will return the following list: {{The} {It's}}

## B.41. lWordLast

Retrieves the last word for all elements of the list.

Synopsis:

**lWordLast** *list*

Description:

This routine takes the given list (list of strings) and extracts the last word from each of the strings. The result value of this routine is a list that contains the last word of each input string.

Example:

lWordLast {{The weather is not fine} {It's raining}}

will return the following list: {{fine} {raining}}

## B.42. lWordNth

Retrieves the *n*th word for all elements of the list.

Synopsis:

**lWordNth** *list n*

Description:

This routine takes the given list (list of strings) and extracts the *n*th word from each of the strings. The result value of this routine is a list that contains the *n*th word of each input string.

If *n* is greater than the number of words in the strings then nothing is returned.

Example:

lWordNth {{The weather is not fine} {It's raining}} 2

will return the following list: {{weather} {raining}}

lWordNth {{The weather is not fine} {It's raining}} 3

will return the following list: {{is} {}}

## B.43. RDNescape

Escapes the characters **=,+;\{}** in the RDN attribute value.

Synopsis:

**RDNescape** *list*

Description:

Escapes the characters **=,+;\{}** in the RDN attribute value. This function is useful before working in Tcl with this variable. After other operations, the escaped character sequences can be removed with the reverse function RDNunescape.

## B.44. RDNunescape

Unescapes the characters **=,+;\{}** in the RDN attribute value.

Synopsis:

**RDNunescape** *list*

Description:

Unescapes the characters **=,+;\{}** in the RDN attribute value. This function is useful after working in Tcl with this variable. The reverse function is RDNescape.

## B.45. replaceAttributes

Returns a variable that contains a REPLACE attribute definition for an LDIF change file.

Synopsis:

**replaceAttributes** *value1 value2 ...*

Description:

Returns a list containing the string REPLACE indicating to replace the following values.

This function is especially valuable when the whole entry must be replaced via an LDIF change file. In this case, all attributes should be handled with the replaceAttributes function.

## B.46. StringAppend

Appends strings to the given string.

Synopsis:

**StringAppend** list *string1 string2 ...*

Description:

This routine takes the given list and appends to all its elements the given strings *string1*, *string2*, ....

The result value of this routine is a list with all the appended values.

Note: The native Tcl command append doesn't work properly for Tcl lists. Therefore the wrapper function is needed.

# B.47. StringModify

Returns a string where either '*n*' or all occurrences of string A are replaced with string B.

Synopsis:

**StringModify** *list string1 string2 ?n?*

Description:

This routine takes the given Tcl list and replaces (*n* times) in each of its strings the substring *string1* with the substring *string2*. If *n* is missing or set to "all", then all of the matching substrings will be replaced.

The result value of this routine is a Tcl list with all the replacements made.

Examples:

StringModify "The weather is not fine" "i" "X" 1

will return the following Tcl string: "The weather Xs not fine"

StringModify "The weather is not fine" "i" "X" 2

will return the following string: "The weather Xs not fXne"

# DirX Product Suite

The DirX product suite provides the basis for fully integrated identity and access management; it includes the following products, which can be ordered separately.



## DirX Identity

DirX Identity provides a comprehensive, process-driven, customizable, cloud-enabled, scalable, and highly available identity management solution for businesses and organizations. It provides overarching, risk-based identity and access governance functionality seamlessly integrated with automated provisioning. Functionality includes lifecycle management for users and roles, cross-platform and rule-based real-time provisioning, web-based self-service functions for users, delegated administration, request workflows, access certification, password management, metadirectory as well as auditing and reporting functionality.



## DirX Directory

DirX Directory provides a standards-compliant, high-performance, highly available, highly reliable, highly scalable, and secure LDAP and X.500 Directory Server and LDAP Proxy with very high linear scalability. DirX Directory can serve as an identity store for employees, customers, partners, subscribers, and other IoT entities. It can also serve as a provisioning, access management and metadirectory repository, to provide a single point of access to the information within disparate and heterogeneous directories available in an enterprise network or cloud environment for user management and provisioning.



## DirX Access

DirX Access is a comprehensive, cloud-ready, scalable, and highly available access management solution providing policy- and risk-based authentication, authorization based on XACML and federation for Web applications and services. DirX Access delivers single sign-on, versatile authentication including FIDO, identity federation based on SAML, OAuth and OpenID Connect, just-in-time provisioning, entitlement management and policy enforcement for applications and services in the cloud or on-premises.



## DirX Audit

DirX Audit provides auditors, security compliance officers and audit administrators with analytical insight and transparency for identity and access. Based on historical identity data and recorded events from the identity and access management processes, DirX Audit allows answering the “what, when, where, who and why” questions of user access and entitlements. DirX Audit features historical views and reports on identity data, a graphical dashboard with drill-down into individual events, an analysis view for filtering, evaluating, correlating, and reviewing of identity-related events and job management for report generation.

For more information: support.dirx.solutions/about

Eviden is a registered trademark © Copyright 2026, Atos Group – All rights reserved.

Legal remarks

On the account of certain regional limitations of sales rights and service availability, we cannot guarantee that all products included in this document are available through the Eviden sales organization worldwide. Availability and packaging may vary by country and is subject to change without prior notice. Some/All of the features and products described herein may not be available locally. The information in this document contains general technical descriptions of specifications and options as well as standard and optional features which do not always have to be present in individual cases. Eviden reserves the right to modify the design, packaging, specifications and options described herein without prior notice. Please contact your local Eviden sales representative for the most current information. Note: Any technical data contained in this document may vary within defined tolerances. Original images always lose a certain amount of detail when reproduced.